MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-12450-TDI/997

PROCESSO DE DESENVOLVIMENTO DE WEB SITES COM RECURSOS DA UML

Isolete Teresinha Dzendzik

Dissertação de Mestrado do Curso de Pós-graduação em Computação Aplicada, orientada pelos Drs. José Carlos Becceneri e Maurício Gonçalves Vieira Ferreira, aprovada em 26 de outubro de 2004.

INPE
São José dos Campos
2005

519.8:303.725.35

DZENDZIK, I. T.
Processo de desenvolvimento de *web sites* com recursos da UML / I. T. Dzendzik. – São José dos Campos: INPE, 2004.
182p. – (INPE-12450-TDI/997).

1.Rede mundial de Computadores (World Web Site-WWW). 2.Websites. 3.Linguagem de Modelagem Unificada (Unified Modeling Language- UML). 4.Desenvolvimento de engenharia. I.Título.

Aprovado(a) pela Banca Examinadora em cumprimento ao requisito exigido para a obtenção do Título de **Mestre em Computação Aplicada.**

Dr. Reinaldo Roberto Rosa

Presidente/INPE, SJCampos-SP

Dr. José Carlos Becceneri

Orientador/INPE, SJCampos-SP

Dr. Maurício Gonçalves Vieira Ferreira

Orientador/INPE, SJCampos-SP

Dr. Airam Jonatas Preto

Membro da Banca/INPE, SJCampos-SP

Dr. Rafael Duarte Coelho dos Santos

Convidado/IEAv-CTA, SJCampos - SP

Aluna: Isolete Teresinha Dzendzik

São José dos Campos, 27 de outubro de 2004.

AGRADECIMENTOS

Agradeço aos meus colegas, professores, orientadores e ao INPE no papel de representante de todas as pessoas que, direta ou indiretamente, contribuíram com a minha ascensão acadêmica.

Meus colegas: que tanto me ensinaram, ajudaram e colaboraram para que eu conseguisse aprovação nas oito disciplinas cursadas. "... E como me senti útil e feliz quando em determinados momentos tive também algo a ensinar e pude colaborar".

Meus professores: Drs. Nilson Sant'Anna, Reinaldo Roberto Rosa, Ulisses Tadeu Vieira Guedes, Antonio Montes Filho, Mauricio Gonçalves Vieira Ferreira e José Carlos Becceneri por terem ministrado suas disciplinas com tal dedicação e didática que mesmo as questões mais difíceis se tornaram passíveis de serem entendidas.

Meus orientadores: José Carlos Becceneri e Mauricio Gonçalves Vieira Ferreira. "Vocês foram as provas mais concretas de que vale a pena investir em nós mesmos, crescer, ser bom, ser feliz. O que está a nossa volta se parece conosco, é o que merecemos! Agradecer é o reconhecimento de Ser agraciado. E como, e quanto me sinto agraciada por perceber que cresci, aprendi muitas lições e me tornei alguém melhor. Se isso não fosse Real não os teria merecido como orientadores".

Ao INPE: pelo espaço físico e por todas as pessoas com as quais tive contato e que de alguma forma, seja direta ou indiretamente, contribuíram para com a minha formação. Contribuições essas que reconheço que existiram na forma de e-mails ou telefonemas informativos, minha entrada na portaria, xerox, empréstimo de materiais, conversas, conselhos, informações, trocas de experiências, festas, opções de alimentação, passeios etc.

RESUMO

O presente trabalho apresenta um processo desenvolvimento de Web *sites* organizado em três fases. Para que haja uma linguagem comum entre cliente e equipe de desenvolvedores, o processo faz uso de termos e diagramas da UML. Este processo não propõe a criação de uma nova tecnologia, mas o uso de diversas tecnologias em conjunto, representando um processo de desenvolvimento chamado Processo de Desenvolvimento *Web* com recursos da UML (PDW-UML). Este Processo é baseado em três fases: a fase de levantamento de requisitos, a de implementação e de *design* das interfaces. A fase de levantamento de requisitos levanta os objetivos do domínio e busca as informações disponíveis para que haja o planejamento do Web *site*; faz o levantamento das necessidades do sistema que são mostradas através dos recursos da UML e organiza os requisitos necessários à implementação. A fase de implementação desenvolve os arquivos e diretórios que formam a arquitetura lógica de acordo com um modelo de Estrutura Dinâmica (ED), proposto para implementar o PDW-UML. Os requisitos levantados na fase anterior são analisados e utilizados para a composição das interfaces e do sistema navegacional. A fase de *design* das interfaces fundamenta-se no uso da arte baseada em técnicas em ferramentas gráficas para *design* na Web e de conhecimentos complementares como da psicologia e da filosofia como auxílio para a realização artística e criativa. A proposta para a fase de *design* é uma forma de cultura que envolve técnicas, arte, estética e ética como uma base de *design*.

WEB DEVELOPING PROCESS WITH UML RESOURCES

ABSTRACT

This work shows a process of website development organized in three phases. To establish a common language between the client and the development team, it makes use of expressions and UML diagrams. This process does not propose the creation of a new technology, but the use of several existing technologies together, in a method named Web Developing Process with UML Resources (PDW-UML). This process is divided into three phases: Requirement Analysis, Implementation and Interface Design. The Requirement Analysis phase starts from the objectives of domain and researches into the available information for planning of the website; it involves assessment of the system requirements as indicated by the UML resources, and planning for implementation. The implementation phase develops the files and directories that define the logical architecture following a Dynamic Structure (ED) model, proposed for implementation of the PDW-UML. The requirements established in the precedent phase are analyzed and used in the composition of the interfaces and of the navigational system. The Interface Design principally involves the use of the art techniques and graphical tools for web design and complementary knowledge such as psychology and philosophy to assist the artistic and creative realization. The purpose of the design phase is to blend technique, art and aesthetics into good design.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

CAPÍTULO 1 - INTRODUÇÃO 19

CAPÍTULO 2 - TECNOLOGIAS INTERNET E *WEB* 23

2.1 - Tecnologias de Rede e de Banda 23
2.1.1 - Rede 23
2.1.2 - Banda 23
2.1.3 - Uso de Banda no Brasil 24
2.2 - Recursos de *Softwares* Aplicados ao Desenvolvimento *Web* 25
2.2.1 - *Softwares* de Desenvolvimento de Páginas *Web* 26
2.2.1.1 - *Softwares* Geradores de Códigos 26
2.2.1.2 - Edição Manual de Códigos 27
2.2.2 - Linguagens de Programação para Páginas Dinâmicas 29
2.2.2.1 - ASP 29
2.2.2.2 - PHP 29
2.2.2.3 - JSP 31
2.2.3 - Linguagens de Programação *Client Side* 32
2.2.3.1 - CSS 32
2.2.3.2 - HTML 33
2.2.3.3 - JavaScript 33
2.3 - Servidores *Web* (Hosts) 34
2.3.1 - Apache 35
2.3.2 - IIS 35
2.3.3 - ChiliASP 36

CAPÍTULO 3 - PRINCÍPIOS, NORMAS E PADRÕES PARA A *WEB* 39

3.1 - Princípios de Usabilidade e de *Design* 39
3.1.1 - Princípios de Usabilidade 39

3.1.2 - Princípios de *Design* 42

3.1.2.1 - Princípios Estéticos 45

3.1.2.2 - Princípios Artísticos 47

3.1.2.3 - Princípios Éticos 50

3.2 - Práticas Recomendadas para a Internet (IEEE 2001) 52

3.3 - W3C 53

3.4 - UML 56

3.4.1 - O uso da Modelagem para Entender Melhor o Sistema 57

3.4.2 - Padrões 57

3.5 - Engenharia de *Web sites* 58

3.5.1 - Problemas no Desenvolvimento de *Web sites* 59

3.5.2 - Diferenças entre a Engenharia de *Software* e a Engenharia *Web* 60

3.5.3 - Principais Atividades 61

3.5.4 - Principais Tipos de Requisitos para *Web sites* 61

CAPÍTULO 4 - ESTRUTURAS E MODELOS DE *WEB SITES* 63

4.1 - Estruturas de *Web sites* 63

4.1.1 - Página Única com *Links* para os Tópicos 63

4.1.2 - *Frames* 64

4.1.3 - Páginas divididas com Tabelas 65

4.2 - Modelos de *Web sites* 65

4.2.1 - HDM 65

4.2.2 - RMM 69

4.2.3 - OOHDM 71

4.2.4 - Conallen 73

4.2.5 - WebML 77

4.2.6 - Tabela Demonstrativa dos Modelos Citados 82

4.2.7 - Modelos Desenvolvidos a Partir dos Modelos Mostrados nesta Seção 84

4.2.8 - Considerações Gerais 84

CAPÍTULO 5 - PROCESSO DE DESENVOLVIMENTO DE *WEB SITES* COM RECURSOS DA UML 87

5.1 - Fase 1: Levantamento de Requisitos 90

5.1.1 - Requisitos de Conteúdo 90

5.1.2 - Requisitos Operacionais 93

5.1.3 - Requisitos de Desenvolvimento 96

5.1.4 - Modelagem do Domínio 98

5.1.4.1 - Interações e Atividades 98

5.1.4.2 - Modelagem das Interfaces Dinâmicas 106

5.1.4.3 - Componentes 112

5.2 - Fase 2: Implementação 112

5.2.1 - Documentação 113

5.2.1.1 - Descrição do Domínio 113

5.2.1.2 - Informações Gerais 113

5.2.1.3 - Convenções 114

5.2.1.4 - Modelagem 118

5.2.1.5 - Atualização e Manutenção 119

5.2.1.6 - Documentação *On-line* 120

5.2.1.7 - Armazenamento de Arquivos Originais 120

5.2.2 - Análise e Seleção dos Requisitos de Conteúdo 121

5.2.3 - Definição do Tipo de Estrutura 126

5.2.4 - Definição dos Níveis dos URLs 132

5.2.4.1 - URLs a Partir de um Único Diretório 133

5.2.4.2 - URLs a Partir do Diretório Respectivo de cada Assunto 136

5.2.5 - Composição das Interfaces 140

5.2.5.1 - Composição dos Arquivos Arquivos-matriz 141

5.2.5.2 - Composição dos Arquivos de *Design* 142

5.2.5.3 - Composição dos Arquivos-menu 143

5.2.5.4 - Composição dos Arquivos de Conteúdo 145

5.2.6 - Testes do Sistema Navegacional 148

5.3 - Fase 3: *Design* das Interfaces 149

5.3.1 - Arte e Técnica como Princípios de *Design* 152

5.3.1.1 - A arte na *Web* 153

5.3.1.2 - A estética na *Web* 156

5.3.1.3 - A ética na *Web* 159

5.3.1.4 - Considerações Técnicas, Artísticas e Éticas em Interfaces Web 160

5.3.1.4.1 - Estrutura de uma Interface 160

5.3.1.4.2 - Coerência e Legibilidade dos Textos 161

5.3.1.4.3 - Uso de Metáfora e Termos Inadequados para a *Web* 162

5.3.1.4.4 - Peso do Montante de Cada Interface 164

5.3.1.4.5 - Harmonia na Distribuição dos Atributos que Compõem uma Interface 164

5.3.1.4.6 - Adoção de um Tipo de Linguagem e/ou Tecnologia 165

5.3.1.4.7 - Objetividade 166

5.3.1.4.8 - Canais de Comunicação 167

5.3.2 - Avaliações e testes 168

CAPÍTULO 6 - RESULTADOS 171

CONCLUSÃO 175

REFERÊNCIAS BIBLIOGRÁFICAS 177

LISTA DE FIGURAS

3.1 - Exemplo de *design* de uma interface 43
4.1 - Exemplo de *links* de uma aplicação derivada 68
4.2 - Tradução de *links* de abstrato para concreto 69
4.3 - Modelo cliente x servidor 73
4.4 - Ícones estereótipos 75
4.5 - Páginas cliente construindo uma página servidora 76
4.6 - Modelo de dados WebML 78
4.7 - Especificação de hipertexto do WebML 79
4.8 - Desenvolvimento de aplicações *Web* 81
5.1 - Visão geral do PDW-UML 89
5.2 - Armazenamento de informações 92
5.3 - Arquitetura física e arquitetura lógica 94
5.4 - Organização hierárquica 95
5.5 - Concentração de custos 98
5.6 - Principais atividades e interações de um *Web site* 99
5.7 - Diagrama de atividades com as ações do cliente 100
5.8 - Diagrama de atividades com as ações da equipe 102
5.9 - Diagrama de casos de uso com as ações dos usuários 104
5.10 - Interfaces estáticas e dinâmicas mostradas em um diagrama de classes 105
5.11 - Casos de uso de interfaces dinâmicas 107
5.12 - Diagrama de classes com o exemplo de um carrinho de compras 108
5.13 - Diagrama de seqüências do caso de uso Gestão de cadastro 109
5.14 - Diagrama de seqüências do caso de uso Fazer Compras 111
5.15 - Diagrama de componentes 112
5.16 - Estereótipo e atributos que compõem as interfaces 126
5.17 - Organização, *script* e *design* de uma interface com ED 127
5.18 - Divisão horizontal com duas células 128
5.19 - Divisão horizontal com três células 129
5.20 - Divisão horizontal e vertical formando três células 131
5.21 - Arquivos-matriz em um único diretório 135
5.22 - Arquivos-matriz em diversos diretórios 147

5.23 - Arquivos-matriz e arquivos-default em diversos diretórios 139
5.24 - Base de inclusão para composição dos arquivos-matriz 140
5.25 - Arquivo-matriz com *meta tags* no cabeçalho 141
5.26 - *Script* com instruções de *design* em CSS 143
5.27 - Arquivo-menu no nível *root* 144
5.28 - Arquivo-menu em um subdiretório 145

LISTA DE TABELAS

2.1 - Comparativo entre tecnologias que suportam páginas dinâmicas 31
2.2 - Cliente x Servidor 33
4.1 - Comparativo entre os modelos apresentados nesta seção 89
5.1 - Informações gerais de um projeto de *Web site* 114
5.2 - Exemplos de referências de cores 116
5.3 - Estereótipos e descrição de interfaces 123
5.4 - Estereótipos dos diretórios 133

LISTA DE SIGLAS E ABREVIATURAS

ASP	-	Active Server Pages
ASP.NET	-	Active Server Pages.NET
CASE	-	Computer Aided Software Engeenering
CGI	-	Common Gateway Interface
CSS	-	Cascading Style Sheets
DHTML	-	Dynamic Hypertext Markup Language
ED	-	Estrutura Dinâmica
E-R	-	Entidade-Relacionamento
ES	-	Engenharia de Software
EW	-	Engenharia Web
FAQ	-	Frequently Asked Questions
FTP	-	File Transfer Protocol
GPL	-	General Public License
HDM	-	Hypertext Design Model
HTML	-	Hypertext Markup Language
HTTP	-	Hypertext Transfer Protocol
IBOPE	-	Instituto Brasileiro de Opinião Pública e Estatística
ID	-	Identifier
IEEE	-	Institute of Electrical and Electronics Engineers
IIS	-	Internet Information Server
JSP	-	Java Server Pages
MCT	-	Ministério da Ciência e Tecnologia
NCC	-	National Computing Centre
OO	-	Object Oriented
OOHDM	-	Object-Oriented Hypermidia Design Method
PC	-	Personal Computing
PDW-UML	-	Processo de Desenvolvimento We-UML
PHP	-	Page Hypertext Preprocessor
RDF	-	Resource Description Framework
RGB	-	Red, Green and Blue
RMDM	-	Relationalship Management Data Model
RMM	-	Relationship Management Methodology
RNP	-	Required Navigation Performance
SEPIN	-	Secretaria de Política e Informática
SGML	-	Generalized Markup Language
SMIL	-	Synchronized Multimedia Integration Language
SSI	-	Server Side Include
SVG	-	Scalable Vector Graphics
UML	-	Unified Modeling Language
URL	-	Uniforme Resourse Locator

W3C	-	World Wide Web Consortium
WAE	-	Web Application Extension
WebML	-	Web Modeling Language
WWW	-	World Wide Web
XML	-	Extensible Markup Language
XSL	-	Extensible Stylesheet Language

CAPÍTULO 1

INTRODUÇÃO

O desenvolvimento de *Web sites* envolve assuntos como a engenharia de *Web sites*, banco de dados *on-line*, metodologias e modelos, técnicas de uso de ferramentas gráficas, técnicas de redação e ainda outros assuntos, conforme necessidades específicas.

As interfaces *Web* podem ser consideradas como estáticas quando o conteúdo é permanente, e dinâmicas quando o conteúdo é gerado a partir de uma consulta a banco de dados, sendo mantido somente durante a permanência do usuário na interface.

Para o desenvolvimento de *Web sites* os profissionais da *Web* fazem uso de linguagens e tecnologias, modelos e metodologias de modelagem e de implementação e de técnicas de uso de ferramentas gráficas e de *design*. Quando um *Web site* é composto somente por interfaces estáticas as necessidades de um projeto são mais centradas na escolha das linguagens de desenvolvimento e nas ferramentas de *design* que serão utilizadas. Quando há interfaces dinâmicas, além da escolha das linguagens e das ferramentas de *design* é necessário fazer a escolha da tecnologia responsável pela geração de páginas dinâmicas; do sistema de banco de dados e de uma linguagem de modelagem que possa proporcionar aos desenvolvedores uma visão do sistema como um todo.

Softwares servidores de páginas dinâmicas como a *Active Server Pages* (ASP), Java *Server Pages* (JSP) e *Page Hypertext Preprocessor* (PHP) têm sido amplamente utilizados para o desenvolvimento de interfaces dinâmicas geradas a partir de uma consulta a banco de dados e para o desenvolvimento dinâmico que se faz pelo aproveitamento de um ou mais arquivos em diversas interfaces usando diretivas de *Server Side Include* (SSI).

A *Unified Modeling Language* (UML) é utilizada no mercado de *software* como uma linguagem gráfica padrão destinada à especificação, construção, visualização e documentação de sistemas de *software*. Os recursos da UML fazem com que esta linguagem seja comum entre desenvolvedores e administradores de *softwares* e *Web sites*. Os *Web sites* são segmentos de *software*, onde os conceitos e as atividades práticas mostram os princípios da interação humano-computador. Isso envolve a aplicação de princípios da engenharia, princípios de usabilidade, técnicas e conhecimentos gerais sobre *design* de interfaces.

Com a disponibilidade de ferramentas fáceis de serem utilizadas, muitos desenvolvedores abusam da liberdade de expressão, criatividade e facilidade de acesso e disponibilizam conteúdos *on-line* sem considerar critérios de usabilidade para a *Web*. Em conseqüência disso começaram a aparecer muitos *Web sites* com problemas resultantes de questões, como:

- A tecnologia de banda não acompanhou a evolução dos recursos de desenvolvimento, com isso usuários de banda estreita têm acesso parcial a conteúdos mais complexos e mesmo usuários de banda larga têm acessos demorados devido a má distribuição de conteúdo por página, muitas vezes desistindo do acesso.
- Há uma dificuldade no entendimento do que é *design,* o que é arte e o que é engenharia. Isso pode gerar *Web sites* difíceis de serem entendidos, feitos sem uma seqüência lógica, sem um projeto prévio, sem ponderação no uso de cores, imagens e textos, dificultando o acesso ao que se busca.
- A falta de consistência de conteúdo e do projeto como um todo, muitas vezes leva a inconsistências entre o objetivo de um *site* e o que se mostra na realidade. Isso pode anular o ideal da *Web* de encurtar o caminho de acesso a informações e levar a perda da credibilidade de uma organização.
- A maioria dos modelos de desenvolvimento *Web* propõe projetos complexos e apenas possibilidades de uso ao invés de uma forma de uso sistematizada. Isso gera dificuldades de utilização por parte dos profissionais da *Web*.

Muitos modelos e metodologias de desenvolvimento para a *Web* já foram apresentados, mas o uso dos mesmos, pelos profissionais da *Web* ainda bastante reduzido (Seção 4.2). A maioria dos modelos propõe soluções que são mais voltadas ao desenvolvimento de *softwares* que para o desenvolvimento de *Web sites*. E os modelos que apresentam soluções para a *Web*, não apresentam uma forma de implementação que possa proporcionar benefícios para os internautas além da compreensão pelos desenvolvedores. Um outro problema encontrado na maioria dos modelos analisados (Seção 4.2) é que estes trazem propostas ou abordagens muito complexas dificultando a utilização pelos profissionais da *Web*.

Há autores que defendem o ideal artístico para a *Web*, outros defendem o ideal da engenharia. Este trabalho pretende mostrar algumas diferenças entre o que é arte do que é engenharia e mostrar onde usar a arte e onde usar a engenharia. Os *Web sites* são “sistemas desenvolvidos

para usuários", então para proporcionar ao usuário um melhor entendimento e um melhor *design*, é necessário que haja o emprego da arte onde se requer arte e da engenharia onde se requer engenharia. Não se trata da criação de uma nova tecnologia, mas do uso formalizado de linguagens e tecnologias para projetar, implementar o *Web site* com uma estrutura padrão e para fazer o *design* das interfaces.

O presente trabalho apresenta um processo de desenvolvimento de *Web sites* voltado ao desenvolvimento de interfaces de *Web sites* chamado de "Processo de Desenvolvimento *Web* com recursos da UML" (PDW-UML). Para que haja uma linguagem comum entre cliente e equipe de desenvolvedores, o processo faz uso de termos e de diagramas da UML e propõe três fases para a efetiva realização, destacando o levantamento de requisitos, a implementação e o *design* das interfaces.

A fase de levantamento de requisitos levanta os objetivos do domínio e busca as informações disponíveis para que haja um planejamento do *Web site*; faz o levantamento das necessidades do projeto e do sistema através dos recursos da UML e organiza os requisitos necessários à implementação. Através da modelagem e da organização do requisitos são mostradas algumas semelhanças e algumas diferenças entre a engenharia de *software* e a engenharia *Web*, respectivamente. Os principais requisitos para *Web sites* são os assuntos que compõem as interfaces. A partir do levantamento dos assuntos e das informações necessárias para compor o sistema de hipertexto (*links* e suas respectivas interfaces) já se pode especificar as necessidades do domínio como um todo e iniciar a implementação.

A segunda fase, que representa a implementação, propõe uma implementação formalizada e um novo modelo de estrutura para as interfaces, fazendo uso da engenharia *Web* utilizando conceitos, recursos de *software* e testes navegacionais. A fase de implementação desenvolve os arquivos e diretórios que formam a arquitetura lógica de acordo com um modelo de Estrutura Dinâmica (ED) proposto para implementar o PDW-UML. Os requisitos levantados na fase anterior são selecionados e utilizados para a composição das interfaces e do sistema navegacional. Nesta fase faz-se a previsão dos níveis de *Uniform Resourse Locator* (URLs) e das portas de entrada do usuário no *site*; fazem-se as convenções de *design* das interfaces; elabora-se a documentação com as informações pertinentes ao *Web site* e as instruções de administração para os *Web designers* e fazem-se os testes navegacionais a partir da máquina servidora.

A partir de convenções de *design* e da implementação do sistema, o *Web site* já apresenta uma estética resultante do trabalho realizado. No entanto, para que se possa aprimorar o *design* é necessária a visualização do sistema como um todo. Para que haja uma atenção aos detalhes e um tratamento ao conteúdo e ao design desenvolvido o PDW-UML propõe uma terceira fase.

A terceira fase, que trata do *design* das interfaces, busca aprimorar o que foi desenvolvido nas fases anteriores propondo um tempo e um espaço dentro de um processo para que haja a realização artística e criativa. A proposta para a fase de *design* é uma forma de cultura que envolve técnicas, estética, arte e ética como base para um estilo de *design*. É na terceira fase que se aprimoram os resultados do trabalho realizado nas fases 1 e 2. Nesta fase o que predomina é a arte e a estética que podem ser vistas na distribuição dos *links* nas páginas, nos títulos, nos cabeçalhos, nos nomes dos *links*, nas cores, nas animações e no *design* em geral. A fase de *design* das interfaces é considerada como artística considerando que após a conclusão das fases 1 e 2 já se obteve a funcionalidade desejada. As duas fases anteriores são baseadas em estudos formais e em soluções já testadas, representando o papel da engenharia. Já nesta fase o resultado pode ser sempre diferente de acordo com a maturidade de cada profissional e do domínio que têm no uso de ferramentas gráficas e de *design* para a *Web* e de conhecimentos artísticos gerais. Isso representa o papel da arte na *Web*.

Independente de o *Web site* ser um sistema simples, com poucos *links* e poucas interfaces e ainda que sejam somente interfaces estáticas o PDW-UML poderá ser utilizado uma vez que abrange um sistema como um todo. Para aplicações mais complexas, com interfaces dinâmicas e uso de banco de dados, o processo representa uma seqüência lógica e prática que pode ser seguida pelos profissionais da *Web*.

Este documento encontra-se organizado em 6 seções. Na Seção 2 encontra-se uma pesquisa sobre conceitos básicos empregados na Internet e na *Web*. Na Seção 3 faz-se uma introdução sobre usabilidade, *design*, princípios, normas padrões, UML e Engenharia. Na Seção 4 encontra-se o estado da arte mostrando alguns tipos de estrutura e de modelos de *Web sites*. Na Seção 5 encontra-se a contribuição deste trabalho através de um processo de desenvolvimento que envolve levantamento de requisitos, implementação e *design* de interfaces. Na Seção 6 mostram-se alguns resultados obtidos.

CAPÍTULO 2

TECNOLOGIAS INTERNET E *WEB*

Nesta Seção apresentam-se as tecnologias que envolvem a Internet, que é a rede mundial de computadores, e a *Web*, abreviatura de WWW (*World Wide Web)*, que é a interface gráfica da Internet disponível através de **servidores de arquivos** (HTML e outras extensões *Web*). Esses servidores são máquinas conectadas à rede para disponibilizar a interface gráfica (Franklint, 2001).

2.1 - Tecnologias de Rede e de Banda

As tecnologias de rede e de banda é que formam a Internet propriamente dita. A seguir são mostrados alguns conceitos sobre rede, banda e uma pesquisa sobre o uso de banda no Brasil.

2.1.1 - Rede

Uma rede é a interconexão entre diversos computadores e outros dispositivos, por meio de cabos, ondas de rádio ou satélite. Uma rede pode ser definida como um grupo de pontos, estações e nós, interligados e o conjunto de equipamentos que os conecta. A transmissão de dados se dá por um canal de comunicação. Há várias modalidades de transmissão: analógica, serial, síncrona e assíncrona. Para que haja uma conexão é necessária a existência dos componentes físicos como os computadores com placas de rede, os cabos, os *hubs*, os conectores etc. Após a instalação dos componentes físicos é necessária configuração das máquinas através de protocolos que habilitem as máquinas para se comunicarem através de uma linguagem comum para que possam trocar dados entre si (Bruce, 2002).

2.1.2 - Banda

Toda vez que se faz uma comunicação entre dois pontos, diz-se que essa comunicação acontece através de um canal ou banda. Por exemplo, quando duas pessoas falam através do telefone comum, elas usam o canal telefônico ou banda telefônica (Drăgănescu, 2003).

Banda é uma faixa de freqüências delimitada, por onde passam as informações de rádio, tv e Internet. O regulamento das telecomunicações reserva bandas diferentes para cada meio de mensagem a fim de evitar a interferência entre os sinais. A largura de uma banda de

freqüências eletromagnéticas está diretamente ligada à rapidez com que os dados fluem. Quanto maior a largura de banda, mais informações podem ser enviadas num dado intervalo de tempo. As bandas são classificadas como banda estreita e banda larga, de acordo com suas respectivas capacidades de transmissão de dados (Bruce, 2002).

2.1.3 - Uso de Banda no Brasil

Ao iniciar um projeto de um *Web site* é necessário considerar um "possível" perfil dos usuários predominantes deste *Web site*, como por exemplo:

- A velocidade de acesso que utilizam em seus computadores;
- Alguns gostos/preferências e hábitos mais comuns;
- A tolerância de espera no carregamento das páginas chamadas.

No Brasil quem faz pesquisas sobre o perfil dos internautas do Brasil e do mundo é o grupo IBOPE *eRatings*. O IBOPE *eRatings* é uma *joint-venture* entre o Grupo IBOPE e a *Nielsen NetRatings*, líder mundial em medição de audiência na Internet. O IBOPE *eRatings* vende os dados e análises aos grandes portais, bancos, agências de publicidade, revistas, que fazem uso destes dados em tomadas de decisão (IBOPE, 2004).

Além do IBOPE *eRatings*, existem outros órgãos como jornais, revistas e empresas especializadas em pesquisas, que fazem levantamentos sobre perfis de internautas.

Algumas considerações extraídas de pesquisas precisam ser levadas em conta para fazer o planejamento de um *Web site* para que as páginas que o compõem tenham o *design* e a velocidade atrativos ao usuário.

- **Considerações sobre as condições de acesso**: uma pesquisa do IBOPE *eRatings* em setembro de 2000 mostra que 70% dos internautas são usuários residenciais e que a maioria usa acesso com o sistema discado e banda estreita ou convencional, onde a velocidade não chega a atingir 56 Kbps (Kilobits por segundo) (SEPIN, 2000). De acordo com um levantamento feito em novembro de 2002, o Brasil já tinha 14,3 milhões de internautas residenciais ativos. Além disso, existem muitos usuários comerciais que também utilizam banda convencional e acesso discado.
- **Número de usuários de banda larga no Brasil em 2004**: segundo informações disponíveis no *site* do Ministério da Ciência e Tecnologia (MCT), o Brasil encerrou

o ano de 2003 com 1,1 milhões de usuários de banda larga, o que corresponde a 6,4% dos 17 milhões de internautas (MCT, 2004).

2.2 - Recursos de *Softwares* Aplicados ao Desenvolvimento *Web*

Os recursos de *software* disponíveis para desenvolvimento de *Web sites* são os mais diversos possíveis que vão desde *softwares* geradores de HTML, JavaScript, CSS (*Cascading Style Sheets*), até os que trazem um servidor de *scripts* de páginas dinâmicas, como o Apache e o IIS (Seção 2.2.2).

O que se deve considerar ao escolher uma forma de desenvolvimento é que um *Web site* não será visto somente pelos usuários na máquina em que foi desenvolvido e sim em uma máquina servidora, a qual nem sempre oferece os mesmos recursos que a máquina de cada usuário. Por isso, é necessário antes de começar a desenvolver um *Web site*, saber em que tipo de servidor ficarão os arquivos e a possibilidade de que um dia poderá ser necessário fazer uma troca de servidor.

As implementações em HTML não costumam apresentar muitas diferenças de um navegador para o outro. Quando se utiliza uma linguagem que precisa dos recursos de um *software* para ser processada, é necessário considerar que a máquina servidora deverá ter a mesma tecnologia que a máquina cliente. Por exemplo, independente do sistema operacional que o desenvolvedor esteja usando, se ele tiver o *Internet Information Server* (IIS) ou o Apache como servidor de páginas dinâmicas, a máquina servidora também precisará ter o IIS, Apache ou semelhante (Br.php.net, 2003).

Os problemas aparecem ao se utilizar linguagens que são processadas diretamente no navegador ou no lado cliente (*Client Side*), porque nem todas as versões ou tipos de navegadores reconhecem todos os recursos de uma linguagem. O mesmo ocorre ao utilizar *softwares* geradores de códigos porque nem todos os servidores reconhecem as formatações geradas por estes *softwares*. Uma saída é conhecer os recursos existentes, seus limites, vantagens, sua relação custo-benefício e testar os resultados dos *scripts* na máquina servidora antes de disponibilizar um *Web site* ao público.

2.2.1 - *Softwares* de Desenvolvimento de Páginas *Web*

Existem duas maneiras de se desenvolver um *Web site*: utilizando *softwares* geradores de códigos ou fazendo edição manual dos códigos, digitando os comandos em um editor de texto e visualizando o resultado através do navegador.

Cada linguagem oferece recursos e algumas funções que uma outra não possui. Por isso é necessário saber quais os recursos que serão necessários a um *Web site* para escolher o que será usado ao invés de fazer uma opção sem critérios por uma linguagem em detrimento da outra. Entre tantas linguagens existentes é difícil determinar qual é a melhor. O que se pode determinar é qual é a melhor para uma aplicação específica. Como não é possível saber com precisão quais recursos existem na máquina dos usuários, uma saída é reduzir ao máximo os recursos que dependem da máquina do usuário, utilizando recursos que dependem da servidora.

Alguns critérios que devem ser considerados ao escolher uma linguagem:

- A linguagem escolhida é suportada pela máquina servidora disponível para a aplicação em questão?
- As seqüências de comandos (*scripts*) destinam-se a máquina cliente ou a servidora?
- Que navegadores ou servidores possuem os recursos que serão necessários para a exibição das páginas? (Microsoft Corporation, 1998).

2.2.1.1 - *Softwares* Geradores de Códigos

Existem centenas de *softwares* geradores de códigos HTML e com extensões ao JavaScript, DHTML, CSS e outras linguagens. Alguns são *freeware* e outros licenciados. Estes *softwares* facilitam a montagem de páginas, pois o usuário trabalha através de instruções disponíveis em cada *software*, como instruções de inserir *link*, imagens, troca de fonte e demais recursos do HTML. As limitações quanto ao uso de *softwares* geradores de códigos são:

- **Reconhecimento de formatações**: na máquina local é possível visualizar o que está sendo desenvolvido, mas quando se faz o envio dos arquivos à máquina servidora as páginas podem não ser exibidas na íntegra, pois os recursos da máquina servidora não conseguem ler todas as formatações. O *Microsoft Front Page*, por exemplo, precisa de um servidor que tenha recursos habilitados para ler suas formatações e

mostrar as páginas como foram desenvolvidas, causando uma dependência de criação e hospedagem em ambiente *Microsoft*.

- **Vinculação de versões de *softwares***: como acontece com outros tipos de *softwares*, quando se usa uma versão mais recente para abrir e dar seqüência em trabalhos que tenham sido feitos em versões anteriores, não há problemas, mas quando se busca o inverso, dependendo da complexidade do documento a versão anterior não reconhece ou mostra parcialmente o que foi feito na versão mais recente.
- **Falta de flexibilidade dos desenvolvedores e administradores**: um profissional que esteja habituado a um determinado *software*, será um especialista naquele *software*, tendo que reaprender a trabalhar em caso de mudança de *software* ou necessidade de programação. Essa falta de flexibilidade gera discussões a respeito de "qual *software* é melhor" ao invés de "o que é melhor para o usuário". Um outro inconveniente no uso de *softwares* geradores de códigos é que o conteúdo das páginas adquire as formatações do *software* que os gerou. No caso de reaproveitamento de conteúdo em outras tecnologias é necessário que os mesmos sejam refeitos.

2.2.1.2 - Edição Manual de Códigos

Na maioria das linguagens é necessário escrever comandos em texto normal (em um editor de texto), coerentes com a sintaxe e com a estrutura da própria linguagem. Estes comandos são popularmente chamados de código fonte, ou simplesmente código. A utilização dos códigos das linguagens é um pouco mais complexa do que a utilização de *softwares* que geram códigos automaticamente, pois através dos *softwares* apenas se escolhe opções e digitam-se os textos, enquanto que o uso dos códigos representa conhecer e digitar os códigos que formam os *scripts* e os textos que formam os conteúdos. Isso exige uma compreensão maior por parte dos desenvolvedores, pois tanto erros de código como de conteúdo podem passar desapercebidos com mais facilidade.

Ao usar edição manual de códigos aumentam as possibilidades de erro de formatação de *tags* e dos textos que representam o conteúdo. Se finalizar ou iniciar uma *tag* ou uma seqüência de *tags* de forma errada ou em um local errado, pode-se comprometer o *design* de uma página e sem corretor ortográfico no editor de textos, a responsabilidade da correção recai sobre quem

está digitando. Uma saída para evitar erros de programação é usar editores inteligentes como o PHP *Editor*, por exemplo, que avisam o usuário através de coloração no código fonte, quando alguma *tag* não está bem formulada (NCC, 1999).

Com a edição manual é possível obter as seguintes vantagens:

- **Simulação de ambientes**: torna-se mais fácil ter instalado na máquina cliente os mesmos recursos que se tem na máquina servidora. Isso evita diferenças de *design* de uma máquina para a outra.
- **Redução de custos**: os custos são referentes ao *hardware* ou ao sistema operacional, quando este é licenciado. Normalmente, o sistema operacional já traz um servidor de páginas dinâmicas, editor de texto e navegador, não havendo necessidade de investir em *softwares* específicos para desenvolvimento de páginas.
- **Aumenta a portabilidade**: permitindo que ao mudar as interfaces ou mudar a tecnologia, seja reaproveitado o conteúdo informativo. Por exemplo, para processar uma página feita em ASP, que é um recurso do Windows, é possível instalar um servidor de ASP no UNIX, no LINUX, bem como é possível instalar um servidor de PHP no IIS. Ainda que fosse necessário mudar de ASP para PHP, por exemplo, toda a estrutura e todo *design* poderiam ser mantidos, trocando somente a linguagem responsável pela estrutura.
- **Aumenta a flexibilidade**: uma vez que se aprende a utilizar uma linguagem e a forma de desenvolvimento, ao trocar de linguagem só é necessário conhecer os comandos da nova linguagem, e o processo de desenvolvimento será o mesmo.
- **Facilita o reaproveitamento do conteúdo quando muda o *design***: a única constante na *Web* é a mudança. Assim, de tempos em tempos torna-se necessário trocar o *design* das interfaces de um *Web site*, porém o conteúdo será o mesmo ou sofrerá apenas alguns acréscimos. Quanto menos instruções de interface estiverem junto aos arquivos de conteúdo, mais fácil será o reaproveitamento e as possibilidades de se ter um *Web site* sempre com *design* atual (Microsoft Corporation, 1998), (NCC, 1999).

2.2.2 - Linguagens de Programação para Páginas Dinâmicas

As linguagens de programação para páginas dinâmicas podem ser utilizadas para o desenvolvimento dinâmico que é feito com as diretivas de SSI ou para a geração de páginas dinâmicas através da consulta a bancos de dados. Esta Seção mostra alguns recursos disponíveis na área de desenvolvimento de aplicações de páginas dinâmicas para a *Web*, como ASP, PHP e JSP.

2.2.2.1 - ASP

A *Active Server Pages* (ASP) é uma tecnologia orientada a objetos, criada pela Microsoft, utilizada para desenvolver páginas HTML dinamicamente. A ASP trabalha com linguagem de *scripts* VBScript baseada no *Visual Basic* da própria *Microsoft*.

Em linguagens de programação *Web*, utilizam-se *tags,* que são "etiquetas" identificadas pelos sinais < e >, que delimitam textos e códigos que sofrerão algum tipo de formatação (Bell, 2000). Na linguagem VBScript os sinais < e > são acompanhados do símbolo % (percentual), tendo portanto, a identificação <% e %>. Para que arquivos com extensão ASP sejam interpretados é necessário um servidor de páginas ASP que interprete os códigos do VBScript que estão entre as *tags* <% e %> e envie ao navegador o HTML gerado (Chase, 2000).

As páginas ASP são desenvolvidas e processadas em Sistemas Operacionais como o Windows NT/ 2000/XP que fornece o IIS ou UNIX/LINUX com o servidor ChiliASP da Chilisoft (Seção 2.3.3).

A tecnologia ASP dispõe do recurso de *Server Side Include* (SSI) que é um processo em que o servidor utiliza as informações de um arquivo e as inclui como parte de outro. A tecnologia ASP já é em si uma espécie de SSI, na qual o servidor utiliza um arquivo HTML e procura por comandos que precisam ser executados e inseridos antes de retornar uma página como resultado dos *scripts* incorporados (CHASE, 2000).

2.2.2.2 - PHP

A PHP (*Hypertext Preprocessor*) é uma linguagem de *script*, *Open Source* de uso geral, interpretada, muito utilizada para o desenvolvimento de aplicações *Web*, podendo ser mesclada dentro do código HTML. A PHP surgiu em 1994 como um projeto pessoal de

Rasmus Lerdorf com o intuito de controlar acessos a sua página *Web*. Atualmente a PHP é um projeto da Apache Software Foundation. A tecnologia PHP incorpora a linguagem PHP que é baseada nas linguagens C, Java e Perl e ainda pode ser vista como uma combinação de linguagem de programação e servidores de aplicações. A linguagem PHP é executada no servidor, sendo enviado para o cliente apenas HTML gerado em uma requisição.

O objetivo principal da linguagem é permitir aos desenvolvedores escrever páginas que serão desenvolvidas ou geradas dinamicamente. Os *scripts* PHP são entendidos e processados pelo servidor de PHP, entre as *tags* :

- **<?php echo ... ?>**: <?php echo("modelo preferencial usado para dispor documentos XHTML ou XML,\n"); ?>.
- **<?echo ... ?>**: <? echo ("modelo mais simples, como uma instrução de processamento SGML\n"); ?>
- <**?= expressão ?>** Uma redução de "<? echo expressao ?>".
- **<script language="php">. . .</script>**: <script language="php"> echo ("processa instruções"); </script>.
- **<% echo ... %>**: <% echo ("para usar *tags* ASP opcionalmente, quando se ativa a diretiva *asp_tags* no arquivo de configuração"); %>.

A linguagem base da PHP é a JavaScript que quando usada fora de um *software* servidor de PHP é uma linguagem interpretada na máquina cliente. Quando utilizada em um servidor de PHP é chamada de **linguagem PHP** e é interpretada no servidor. A PHP tem como principais características (Br.php.net, 2003):

- **Código Aberto**: todo o código fonte da PHP está disponível, basta ir ao *Web site* oficial e fazer o *download*..
- **Multiplataforma**: a PHP pode rodar sobre o Unix, Linux ou Windows.
- **Acesso a Bancos de Dados**: acessa diretamente os principais bancos de dados utilizados atualmente, como dBase, Interbase, MySQL, Oracle, SyBase, PostgreSQL, e outros (Br.php.net, 2003).

2.2.2.3 - JSP

A *Java Server Pages* (JSP) é baseada na linguagem Java, criada pela *Sun Microsystens*, para simplificar o processo de desenvolvimento dinâmico de *Web sites*. A JSP funciona como um compartimento que incorpora elementos dinâmicos. A JSP é uma linguagem de *script*, compilada, que funciona no lado do servidor, ou seja, as páginas *JavaServer* são arquivos texto, normalmente com a extensão ".jsp" que substituem os arquivos estáticos HTML. As páginas JSP, além de utilizar objetos do servidor, podem incorporar e manipular objetos próprios, como *Applets* e *Servlets*.

A JSP é um sistema híbrido de *templates*, parecido com ASP e PHP. A linguagem padrão utilizada nas JSP's é Java puro, mas qualquer tipo de linguagem de *script* pode ser utilizada no lugar do Java, como ASP, PHP, XML e ColdFusion. As novas especificações da JSP 1.1 implementam *tags* especiais que substituem o código Java puro dentro da página JSP. Os *servlets* e a JSP, em 1998, tiveram suas primeiras versões incorporadas ao Java *Web Server*.

Antes mesmo da *Sun* lançar a versão JSP 1.1, a parceria com a Apache gerou o Projeto *Jakarta*, permitindo a integração dos *softwares* de ambas as empresas. Isso resultou em um projeto que tem como objetivo desenvolver um código aberto para implementar aplicações Java no servidor mais utilizado do Linux e Unix, o Apache Server (Medeiros, 2002).

Para melhor entender as tecnologias citadas na Seção 2.2.2, mostra-se uma tabela comparativa, destacando as principais características.

TABELA 2.1 - Comparativo entre tecnologias que suportam páginas dinâmicas.

Características	**ASP**	**PHP**	**JSP**
Arquitetura	Fechada	Aberta	Aberta
Uso de *scripts*	VBScript e JScript	JavaScript	
Segurança	Baseada na arquitetura do NT	Versatilidade de configuração de segurança de acesso	Modelo de segurança do Java
Acesso à base de dados	ADO	DBASE, Interbase, MySQL, Oracle, PostgreSQL	JDBC
Personalização de *tags*	Não pode ser ampliado	Não pode ser ampliado	Ampliado através do uso de biblioteca

FONTE: Medeiros (2002).

O uso de uma ou de outra linguagem se dá de acordo com a experiência de cada profissional e com prioridades e necessidades de cada projeto.

2.2.3 - Linguagens de Programação *Client Side*

As linguagens de programação para a *Web* são os *scripts* que compõem os códigos ou seqüências de códigos, que formam os arquivos existentes em um *Web site*. Algumas linguagens são interpretadas na máquina cliente (*Client Side*) onde o conteúdo gerado é exibido conforme os recursos disponíveis em cada navegador. Se o navegador não tiver os recursos que estão nos *scripts*, o usuário será privado de visualizar parte do conteúdo. As linguagens interpretadas em um servidor (*Server Side*) dependem do servidor para interpretá-las, assim quando um usuário faz uma requisição, a servidora processa os *scripts* que compõem a página e devolve ao cliente somente o resultado na forma de HTML. (Mendes,1999), (Haddleton, 1997).

A Tabela 2.2 mostra algumas características de máquinas cliente *versus* servidoras.

TABELA 2.2 - Cliente x Servidor.

Cliente	Servidora
Quem quer a informação	Quem tem a informação
Usuário	Fornecedor
Quem quer usar	Quem tem o recurso
É um intermediário	Continuamente, escuta e atende os pedidos de clientes
Traduz as solicitações	Fornece hiperdocumentos disponíveis em seu acervo
Exibe hiperdocumentos	Processa códigos de páginas ativas e devolve ao cliente em forma de hiperdocumentos

FONTE: Haddleton (1997).

Algumas linguagens de programação *client side*, amplamente utilizadas para a *Web* são mostradas a seguir.

2.2.3.1 - CSS

A *Cascading Style Sheets* (CSS) foi introduzida quando do lançamento do navegador Internet *Explorer* 3. Logo em seguida, quando lançou a versão 4 do Netscape Communicator, a Netscape também aderiu a esse padrão. Porém quando a Microsoft lançou a versão 4, ela já agregou várias funcionalidades novas, que entretanto se mostraram incompatíveis com o Netscape. Após o lançamento do Internet Explorer 5, com uma carga enorme de funcionalidades para a CSS, as diferenças cresceram, e com elas mais problemas de incompatibilidade entre navegadores.

Por isso, para utilizar CSS é necessário testar em diversos navegadores, os recursos que serão implementados em um *Web site*, para permitir que os usuários não deixem de ver o conteúdo

e os que tiverem as versões mais recentes de navegadores ainda o vejam de uma forma mais completa.

As folhas de estilo facilitam o trabalho de um *Web design* quando se trata de garantir uma formatação homogênea e mais criativa por todas as páginas de um *Web site* e ainda permite mais interatividade com o usuário. Mesmo que se deseje mudar todo o *design*, ou um certo grupo de formatação, as folhas de estilo permitem que uma alteração possa repercutir em todas as páginas do *Web site*.

As folhas de estilo podem ser comparadas a um gabarito de formatação de páginas HTML. Ela permite que se alterem quase todas as *tags* da linguagem HTML. Sua limitação está na falta de reconhecimento por algumas versões de navegadores, sendo utilizada, portanto, como uma linguagem complementar ao HTML.

As instruções de CSS são inseridas entre as *tags* <STYLE > e </STYLE>. Basta que se insira uma vez a *tag* no código para que toda a página responda às instruções (Bos, 2003).

2.2.3.2 - HTML

A linguagem HTML foi criada com o objetivo de dar à rede mundial de computadores um aspecto gráfico. Até a criação de HTML, as ferramentas existentes, tais como *ftp*, *gopher* e *telnet*, rodavam em terminais de caracteres e, apesar de muito úteis e bastante populares no meio acadêmico, eram muito pouco atrativas para o grande público.

A HTML, em conjunto com o protocolo *HyperText Transfer Protocol* (HTTP) e com os navegadores, foi a responsável pela popularidade da Internet. Não se trata de uma linguagem de programação propriamente dita, mas de uma linguagem de formatação, que define um conjunto de *tags* que afetam a maneira como os documentos são exibidos pelo navegador. A HTML consiste em uma linguagem de descrição de textos que é usada como padrão internacional para formatação dos documentos na *Web*, na forma de aplicação de *Standard Generalized Markup Language* (SGML). Para trabalhar com HTML, utiliza-se um editor de texto e um navegador para visualização (Franklint, 2001).

2.2.3.3 - JavaScript

Os *scripts* escritos em JavaScript podem ser colocados dentro das páginas HTML, pois se trata de uma linguagem de *script* que é processada diretamente no navegador, dispensando a

ajuda de um servidor. Ao contrário da HTML que é uma linguagem estática, com a JavaScript se fazem animações com textos e imagens e diversas interatividades com usuários, sendo assim considerada um acessório da HTML. A linguagem JavaScript possibilita o uso de diversos objetos na composição de uma página. Todos eles possuem propriedades que podem ser alteradas, além disso, os objetos fornecem eventos que possibilitam que uma página execute uma ação conforme instrução de um usuário. JavaScript é uma linguagem estruturada que usa objetos, mas não é uma linguagem de programação orientada a objetos. Os objetos são usados para representar algum aplicativo. Sua utilização requer um editor de texto e um navegador para visualização (NCC, 1999).

2.3 - Servidores *Web* (*Hosts*)

Um servidor é um programa que provê algum tipo de serviço para outros programas, denominados clientes. A conexão entre clientes e servidores é implementada normalmente através de passagem de mensagens, por meio de uma rede, utilizando algum protocolo para codificar as requisições dos clientes e as respostas do servidor (Haddleton, 1997).

Um *Web site* é um conjunto de documentos ou páginas com padrão *Web*, que pode ser acessado através da rede mundial Internet. Estes documentos têm um endereço próprio (também chamado domínio) que está localizado na máquina servidora *Web*. No Brasil este endereço (domínio) é autorizado/fornecido pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Existem muitos tipos de servidores *Web*, com várias alternativas de recursos, inclusive gratuitos, no Brasil e no Exterior. Servidor é a denominação mais comum dada a um computador permanentemente conectado à Internet, de forma que usuários possam acessar os arquivos desses servidores, bastando fazer uso de um computador conectado à Internet.

As páginas disponíveis na Internet permitem vários tipos de interatividades com o usuário. As páginas feitas basicamente em HTML oferecem pouca interatividade. Sua interatividade limita-se em clicar em *links* para abrir uma nova página, copiar conteúdos e fazer *download* de arquivos (Chase, 2000).

Linguagens como CSS e JavaScript oferecem auxílio ao HTML para que as páginas tenham um pouco mais de interatividade e movimento. Porém a CSS não é reconhecida por todos os navegadores e o JavaScript, pode ser desabilitado dos navegadores, pelos usuários.

Devido a essas limitações é que foram criados os servidores de páginas dinâmicas, que são *softwares* que trazem recursos para interpretar determinados códigos e extensões de arquivos. Com isso o usuário não fica privado dos recursos utilizados no desenvolvimento, pois é o *software* servidor de páginas dinâmicas que se encarregará de processar os arquivos com as extensões por ele reconhecidas e devolver ao usuário o HTML gerado. Isso possibilita o desenvolvimento dinâmico e mais interatividade nas páginas, independente do navegador, conforme mostrado na Seção 2.2.3.

Destacam-se a seguir alguns servidores *Web*.

2.3.1 - Apache

O Apache é um *software* livre (GPL) que funciona como um servidor *Web*, tanto no Linux como no Windows. O Apache é um projeto de desenvolvimento colaborativo com objetivo de desenvolver o melhor servidor *Web* em performance, robustez, flexibilidade e com padrões de excelência e qualidade. O Apache tem em seu grupo de trabalho programadores das Universidades MIT, Berkeley, Stanford e empresas como IBM, Sun, HP, Compaq, RedHat e diversas outras.

Entre suas principais características está multi-plataforma, robustez, performance, adaptabilidade, gratuidade e boa documentação. Ele tem vantagens em cima dos outros servidores, como código fonte completo e uma licença irrestrita. É compatível com a especificação HTTP/1.1, permite mudanças em suas características mesmo as mais internas através da utilização de módulos (isso implica em flexibilidade) e tem sua própria API padronizando toda programação interna. (The Apache Software Fondation, 2002).

2.3.2 - IIS

O *Internet Information Server* (IIS) faz parte da "família" Microsoft Windows Server que é uma integração de servidores local e *Web*. O IIS é um *software* para criação e hospedagem de aplicações *Web* dinâmicas, gerenciamento de FTP e SMTP. O IIS é responsável pelo processamento das páginas ASP, podendo também processar páginas PHP, desde que este recurso seja instalado (Microsoft Corporation, 2003).

2.3.3 - ChiliASP

Com o lançamento do *software* ChiliASP da empresa ChiliSoft, para rodar em servidores Unix, Netscape e Lotus, a Microsoft deixou de ser a única responsável pelo processamento e hospedagem de páginas ASP. A versão do ChiliASP foi sendo ampliada, primeiro para os servidores Netscape e Solaris e mais tarde para outras versões do Unix (incluindo Linux). A ChiliSoft não foi contratada pela Microsoft para oferecer estes serviços, mas a Microsoft também não foi totalmente contra as realizações da ChiliSoft em expandir a tecnologia *Microsoft* para outros servidores.

O ChiliASP oferece suporte completo a ASP, incluindo objetos como *Session* e *Request*, mesmo sendo a Microsoft, a proprietária dessa tecnologia. Isso significa que é possível usar uma base Unix para desenvolver e processar scripts ASP.
O que ocorre é que está havendo uma certa resistência dos profissionais da *Web* em aceitar trabalhar com a tecnologia Unix ASP. A questão é sobre até que ponto é viável utilizar uma tecnologia base do Windows em um ambiente Unix. Uma possível resposta referente a plataforma Windows é que há um número vasto de ferramentas de desenvolvimento e desenvolvedores experientes que usam essas ferramentas. Muitas empresas querem utilizar as ferramentas disponíveis no Windows e a experiência que têm ao trabalhar com serviços de Internet e Intranet, o que não significa que aplicativos desenvolvidos em Windows tenham que ser mantidos no IIS e no Windows NT/2000. Há empresas que preferem usar a praticidade e a versatilidade do Windows e o desempenho, segurança e estabilidade do Unix como servidor, de Internet e Intranet. Além disso, muitas tecnologias do Windows poderiam ser utilizadas no servidor Unix, como o desenvolvimento de aplicações com VBScript e ASP, sem a necessidade de um desenvolvedor ter que aprender a programar em uma nova linguagem ou usar uma nova tecnologia (Sun.com, 2003), (Yager, 1999).

O que se pode afirmar que é constante na *Web* são as mudanças. Se por um lado isso faz com que novos recursos, linguagens e ferramentas sejam disponibilizadas, por outro lado traz dificuldades para que os profissionais conheçam o que há disponível. Na maioria das vezes é difícil adotar uma linguagem, ferramenta ou servidor de páginas dinâmicas e considerar como sendo a melhor solução para todas a situações. Há profissionais que preferem fazer uso de codificação manual, outros preferem usar ferramentas de design e codificação. Para um usuário dos sistemas UNIX/LINUX, uma solução pode ser a utilização de linguagens, editores e demais ferramentas que melhor se adaptem aos servidores utilizados por estes

sistemas. Uma outra questão que pode auxiliar na escolha de recursos é fazer uma análise das necessidades específicas de cada domínio e encontrar as melhores soluções para tais necessidades.

CAPÍTULO 3

PRINCÍPIOS, NORMAS E PADRÕES PARA A *WEB*

Nesta Seção mostram-se alguns princípios, normas, padrões e recursos da UML e da Engenharia *Web* aplicados ao desenvolvimento de *Web sites*.

3.1 - Princípios de Usabilidade e de *Design*

Muito se discute sobre o que são questões de usabilidade na *Web* e o que são questões de *design*. Nesta Seção são mostrados alguns conceitos e exemplos que visam mostrar características pertinentes à usabilidade e ao *design*. Reconhece-se também que há questões que podem ser consideradas em ambos os casos como, por exemplo, onde uma opção de *design* pode melhorar ou comprometer a usabilidade (contraste entre cor de fundo e cor de fonte é uma opção de *design* que pode interferir de forma positiva ou negativa na usabilidade).

3.1.1 - Princípios de Usabilidade

Usabilidade significa "facilidade de uso". Alguns princípios de usabilidade desenvolvidos para serem usados pela engenharia de *software* também são utilizados pela engenharia *Web*.

A engenharia da usabilidade é baseada em vários segmentos como, por exemplo, a psicologia cognitiva, a sociologia, a ergonomia, a semiótica e a engenharia de *software*. A usabilidade é representada por um subsistema do *software* interativo cujos componentes e processos analisam a interação com seus usuários. Assim um único sistema de interface humano-computador permite várias interações humano-computador, cada uma associada aos diferentes percursos (processos) realizados pelos diferentes usuários (Leite, 2002).

A usabilidade considera algumas questões a serem formuladas e de acordo com as respostas obtidas pode-se ter uma noção do nível de usabilidade de um sistema.

- **Contexto**: onde o sistema interativo será empregado, incluindo componentes como usuários, tarefas e ambiente (equipamento, instalações, iluminação, ruído, organização, interrupções, restrições etc.).

- **Problemas**: Onde encontrar o que se procura? Como solicitar esse serviço? Quais informações devem-se fornecer? Qual o resultado? Obteve-se o resultado esperado? Para que serve esse elemento? O que significa essa figura? Para onde leva esse *link*?
- **Facilidade de aprendizado**: interação com o sistema de forma natural; facilidade de utilização; interface preparada para adaptar-se ao nível de conhecimento e habilidade dos usuários (*wizards* para os inexperientes e teclas de atalho para os mais experientes); ser intuitiva; comandos claramente visíveis para evitar memorização.
- **Diálogo simples e natural**: expressões e conceitos do conhecimento do usuário; evitar termos técnicos da computação; evitar informações irrelevantes; *feedback* ao usuário; mecanismos para informar comportamentos do sistema como localização e tempo de execução.
- **Clareza na arquitetura da informação**: o usuário consegue discernir o que é prioritário e o que é secundário no *site*; informação deve estar estruturada e bem localizada; um mapa do *site* pode ser muito útil.
- **Facilidade de Navegação**: navega-se com facilidade quando um usuário chega à informação desejada em até três níveis; as informações são bem distribuídas e os *links* são representativos.
- **Simplicidade**: quanto mais rápido consegue-se chegar até a informação desejada, melhor é; evitar adornos desnecessários, sem prejudicar o enfoque da aplicação.
- **Relevância do conteúdo**: conteúdo relevante e apropriado para a *Web*; textos concisos e com credibilidade; informações relevantes devem ser apresentadas nas páginas principais; informações secundárias devem ser disponibilizadas em páginas de suporte e conectadas através de *links*.
- **Manter a consistência**: um mesmo padrão deve ser sempre adotado; mesmo que o conteúdo mude com freqüência, característica relevante em aplicações hipermídia, o usuário terá facilidade em lidar com a aplicação, pois já irá conhecer os procedimentos padrões.
- **Foco no usuário**: princípio que reúne todos os demais. A usabilidade orienta as aplicações para que foco esteja nas atividades dos usuários.

- **Simbologia**: ao se falar de usabilidade na *Web*, não se pode deixar de levar em consideração que se trata de uma rede mundial, portanto deve-se ter uma preocupação com o processo de internacionalização. Deve-se ter em mente, que somente o uso de interfaces gráficas ou uso de elementos gráficos ao invés de palavras não resolve grande parte dos problemas, já que alguns símbolos podem ter interpretações distintas. É necessário que o projetista de IHC (Interface Humano-Computador) tenha, no mínimo, consciência de que a usabilidade de sua interface não pode ser dimensionada apenas pelos conhecimentos técnicos de sua área específica de atuação. Usabilidade compreende uma gama de diretivas de diversos ramos de atuação (Leite, 2002).

Em alguns pontos a engenharia de *software* é muito semelhante à engenharia *Web* e conseqüentemente as questões de usabilidade podem ser úteis a ambas. A usabilidade para *Web sites* ainda está sendo moldada e muitas questões mostradas como princípios de usabilidade são apenas perguntas ao invés de respostas que poderiam ajudar os profissionais da *Web* a terem uma base melhor. E há muitas questões que são relevantes para a engenharia de *software*, mas não são para a engenharia *Web*. Quando uma equipe desenvolve um sistema de *software* e implementa dentro de uma empresa, normalmente, realiza-se um treinamento com os funcionários, mostrando como usar o *software* de forma mais intuitiva. Já quando se disponibiliza um *Web site on-line* não há como treinar os usuários que farão uso. Assim, o que pode ser intuitivo para usuários de um *software* local pode não ser para usuários da *Web*.

Em várias passagens deste trabalho menciona-se o termo "intuição e intuitivo". Uma maneira de entender melhor o significado e funcionamento é seguir o exemplo mostrado pela psicologia cognitiva para ilustrar a origem do conhecimento humano e as formas de inteligência (Seção 3.1.2.2) utilizadas para fazer uso dos conhecimentos adquiridos.

A origem etimológica da palavra intuição provém de um verbo latino que significa ver; perceber; discernir; ato ou capacidade de pressentir. É uma forma de captar informações sem recorrer aos métodos do raciocínio e da lógica. A intuição não se opõe à razão, apenas há mais probabilidade de acerto quando intuição e razão agem de forma equilibrada, ou quando é possível explicar uma intuição através da razão (Theóphilo, 2003).

Quanto à questão das interfaces intuitivas, existe uma grande diferença entre interfaces de *softwares* convencionais e interfaces de sistemas *Web*. O problema é que autores como,

Jakob Nielsen (Nielsen, 2000), por exemplo, propõem questões de usabilidade para sistema de *softwares* e sugerem os mesmos princípios de usabilidade para sistemas *Web*. Assim o que poderia ser intuitivo em um sistema de *software* (em que os mesmo usuários fazem uso diariamente) também seria intuitivo para um sistema *Web* (onde não há como conhecer o perfil dos usuários que podem fazer uso do *Web site* diariamente, periodicamente ou uma única vez).

Quando uma equipe desenvolve um *software* para uma empresa, normalmente esta equipe treina os usuários, mostrando o que cada ícone representa. A partir daí, pode-se deduzir que a prática poderá fazer com que o sistema seja utilizado sem grandes esforços racionais, ou seja, de forma mais rápida e intuitiva. No entanto para usuários da *Web*, o que se pode propor como procedimentos intuitivos são questões baseadas no sistema cognitivo humano e não em "intuições" aproveitadas da engenharia de *software* (sem uma base racional e sem considerar a diversidade de usuários de sistemas *Web*), como proposto até agora pelo chamados "gurus" da Internet.

Não se pode negar o poder da semiótica, mas o entendimento dos símbolos difere de um local para outro, de uma pessoa para outra, de uma faixa etária para outra. O uso de imagens em interfaces *Web* é de extrema importância, mas quando tem o objetivo fortalecer a afirmação de um texto ou de um fato. E não simplesmente por uma "crença" de que os usuários saberão qual foi o objetivo do desenvolvedor e fará uso conforme sua determinação.

Hurlburt faz a observação de que Confúcio disse que "*uma imagem vale mais que mil palavras*", no entanto usou palavras para dizer isso (Hurlburt, 2003). Este é um exemplo de que nem tudo o que vale para um *software* implementado dentro de um espaço fixo, previsível, vale para a um sistema *Web*, implementado em um servidor *Web* onde não se pode medir sua dimensão, treinar usuários, receber *feedback* de todos os usuários ou conhecer seus recursos de *software*, *hardware* e banda etc. Isso quer dizer que os princípios de usabilidade podem ajudar muito um desenvolvedor de sistemas quando este os utiliza conforme os devidos fins.

3.1.2 - Princípios de *Design*

O *design* é a parte visível de uma interface, é o desenho, a forma, as cores, os alinhamentos etc., sem que haja um julgamento ou uma tentativa de entender ou dar nome ao que se vê. A partir do momento em que se procura entender um *design* e atribuir nomes ao que aparece em

uma interface, faz-se referência ao conteúdo ou mais precisamente, aos atributos que compõem uma interface.

Para ilustrar o conceito de *design* basta acessar um *Web site*, cujo idioma não se tem nenhum conhecimento como, por exemplo, um endereço japonês, chinês etc. visitado por quem não conhece o idioma, tendo condições somente de ver o *design*.

A Figura 3.1 mostra uma interface de um *Web site* japonês, considerando que o usuário não entenda este idioma e que possa somente ver o desenho sem entender o que o desenho diz.

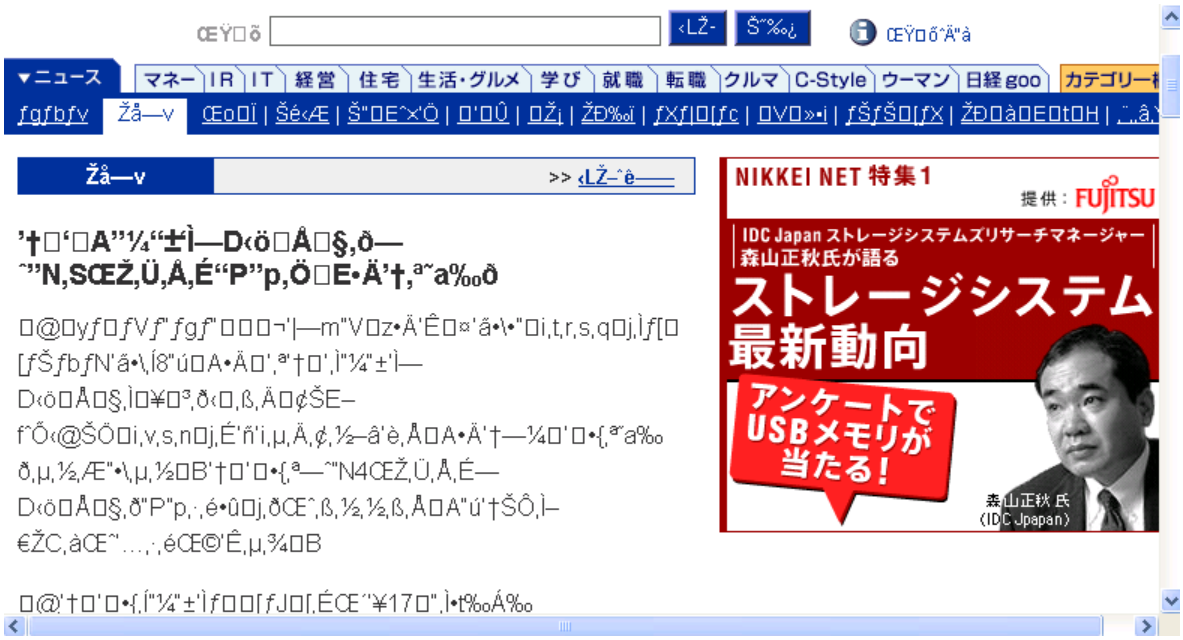


FIGURA 3.1 - Exemplo de *design* de uma interface.
FONTE: Shimbun (2005).

Design é desenho, esboço, projeto. Pode-se considerar como *design* de um *Web site*, tudo o que visualizamos em uma página, ou seja, os textos e as imagens, as cores, os alinhamentos e as figuras geométricas como tabelas e linhas. Pode-se dizer que um texto também é *design* quando não há um julgamento sobre o seu significa ou quando não há uma tentativa de fazer uma leitura. Quando olhamos para o conteúdo visual de um texto podemos dizer que estamos olhando o *design*.

O *design* muitas vezes é confundido com estrutura, *layout*, *design pattern*, *template*, arte etc. Estes termos (embora não haja uma tradução exata para *layout* e *design*) têm conceitos semelhantes para determinadas situações, mas na prática cada um representa situações que têm características diferentes.

- **Estrutura**: divisões, compartimentos, espaços vazios aptos a receber conteúdos; "diagrama" padrão de interfaces *Web* que exerce um papel de "recipiente" de atributos (Seção 4.1).

- ***Layout***: disposição ou posição que os atributos ocupam em uma interface. Este termo é pouco usado para fazer referências a interfaces *Web* uma vez que é válido somente para especificar esquemas conceituais de posicionamento dos atributos. O termo *layout* não é considerado elegante para especificar um processo de *design*. Profissionais que trabalham com artes gráficas preferem ser conhecidos como diretores de arte gráfica, *Web designer*, diretor de *design* ou comunicadores visuais, em vez de *layoutmen* ou *layouter* (Hurlburt, 2002).

- ***Design pattern***: O *design pattern* tem suas raízes no trabalho de Christopher Alexander, um engenheiro civil que escreveu sobre a sua experiência em resolver problemas de projeto que ele encontrava em construções (Seção 3.5.2). Ocorreu a Alexander que certas idéias de projeto, sempre que eram utilizadas, levavam ao efeito desejado. *Design patterns* são representados como relacionamentos entre classes e objetos com responsabilidades definidas que agem em harmonia a fim de levar a uma solução. A documentação de *design patterns* é altamente estruturada. Os padrões são documentados a partir de um modelo que identifica a informação necessária para entender o problema do *software* e a solução em termos de relacionamentos entre as classes e objetos necessários para implementar a solução. Não existe um consenso dentro da comunidade de *design patterns* sobre como descrever um *template* de *patterns*. Diferentes autores preferem diferentes estilos para seus *templates* de *patterns* (Mesquita, 2003).

- ***Template***: é uma forma de documento estruturado que captura as informações essenciais necessárias para o entendimento da essência de um problema e a estrutura da solução. O termo *template* também é utilizado para fazer referências a "modelos de *design*" de *Web sites*.

- **Arte**: conhecimentos e habilidades que caracterizam um trabalho como não sendo puramente técnico, mas que são resultados do uso de técnicas, ferramentas, materiais e criatividade (Domingues, 2003), (Chauí, 2000), (Aranha, 2003), (Hurlburt, 2002), (mais detalhes na Seção 3.1.2.2).

Design não é apenas algo relacionado à beleza, porém a beleza está relacionada ao *design*. Também há outros fatores tais como: técnicas, ergonomia, materiais, tecnologia, adaptação, mercadologia, metodologia, usabilidade, aceitação no mercado etc. *Design* é desígnio, é designar coisas, projetar qualidade, tendências, inovar ou renovar.

Na concepção da palavra, *design* gráfico é uma arte aplicada e funcional que se encarrega de melhor organizar visualmente espaços e informações visuais, facilitando ao máximo a compreensão do espectador, tornando visualmente agradável e interessante essa "leitura" e buscando sempre a inovação e evolução dessa forma de arte. Harmonia, clareza, limpeza e beleza são características importantes de um trabalho bem feito em todas as áreas e em interfaces *Web* não poderia ser diferente (Hurlburt, 2002).

- **Histórico**: *design* é uma palavra ambígua. No século XVIII na Inglaterra, o termo significava “plano de uma obra de arte”. Na origem latina, “designare” significa simultaneamente “a idéia de desenho e desígnio e implica o conceito de um objeto em vias de produção”. *Design* então é definido tanto como um projeto ou o produto de um planejamento (Hurlburt, 2002), (Domingues, 2003).
- **O profissional *designer***: tem a visão do mercado consumidor, seu trabalho deve gerar retorno para o cliente e atrair o consumidor aplicando conhecimentos e recursos de arte. Esses conhecimentos não o qualificam como artista, mas como um *designer* com conhecimentos técnicos aprimorados.
- **O artista**: expressa idéias e sentimentos, sendo que o artista que faz a “arte pela arte” não precisa se preocupar com quem terá contato com sua obra (Aranha, 2003). O artista não transita por vários estilos. Já o *designer* em seu processo de desenvolvimento é capaz de transitar por vários estilos para elaborar a melhor solução (Domingues, 2003).

3.1.2.1 - Princípios Estéticos

A estética está presente em interfaces de *Web sites* tanto no conjunto como um todo, como nos detalhes estilizados que caracterizam cada atributo (texto, *link*, imagem, formulário). Há uma estética básica para a composição de interfaces que é baseada nas cores branca (fundo), preta (textos), cinza (formulários), azul (*links*) e alinhamento à esquerda. No entanto, as possibilidades de modificações das características da interface básica, bem como dos

atributos que a compõem é que faz com que as interfaces tenham estéticas diferentes ou *designs* diferentes.

Dificilmente um *Web designer* mudará as características originais dos atributos com a "intenção" de torná-los menos atraentes ou menos belos. Um exemplo disso são os cursos de *Web design*, intensivos, de ensino médio e universitário que são aplicados com a finalidade de proporcionar a um profissional, conhecimentos para criar *designs* diferenciados a partir da criação, modificação e posicionamento dos atributos que compõem uma interface.

Estética é uma área da filosofia que estuda racionalmente o belo e o sentimento que o belo desperta nos seres humanos.

Etimologicamente, a palavra estética vem do grego *aisthesis*, com o significado de faculdade de sentir, compreensão pelos sentidos, percepção totalizante. A ligação da estética com a arte é ainda mais estreita quando se considera que o objeto artístico é aquele que se oferece à percepção. Por isso pode-se compreender que, enquanto disciplina filosófica, a estética tenha também se voltado para as teorias da criação e da percepção.

A recepção estética é a experiência estética que não visa o conhecimento lógico, medido em termos de verdade, não visa a ação imediata e sua utilidade não pode ser julgada para um determinado fim. A experiência estética é a experiência da presença tanto para o objeto estético como para o sujeito que o percebe (Aranha, 2003).

No entanto, a maneira como a estética é interpretada ou julgada pode variar de pessoa para pessoa e a "psicologia cognitiva" mostra algumas razões dessas diversidades. Segundo Marilena Chauí, não há diferença entre sensação e percepção. A percepção é uma relação do sujeito com o mundo exterior e esta relação dá sentido ao percebido e ao percebedor, e um não existe sem o outro. O ser humano sente e percebe formas, isto é, totalidades estruturadas dotadas de **sentido e significação**. A percepção envolve a personalidade, a história pessoal, as afetividades, por isso uma experiência estética pode ser positiva ou negativa conforme o julgamento que se faz dela (Chauí, 2000).

A ação de "olhar para a estética de uma interface" é comum a todos, mas o julgamento que ocorre a partir dessa ação é que é peculiar a cada usuário fazendo com que os usuários sejam mais ou menos exigentes, de acordo com seus conhecimentos, gostos, preferências. Isso mostra uma razão pela qual os profissionais da *Web* buscam novas maneiras de desenvolver e apresentar uma estética aos usuários fazendo alteração das formas primitivas dos atributos

das interfaces e a criação de novas formas de ilustrações. Estas criações envolvem cores, tamanhos, alinhamentos, movimentos etc.

A palavra estética é usada em vários sentidos, porém, todos partem do conceito primitivo usado pelos gregos antigos: designar aquilo que tem a ver com os sentimentos, com os sentidos, com a percepção. Assim, a estética também está ligada à atividade artística, já que se preocupa com as obras que o ser humano faz com a finalidade de serem belas, e com reações que elas provocam. Em termos gerais, pode-se dizer que a estética é a área da filosofia que estuda a arte e suas relações com o ser humano (Gallo, 1997).

3.1.2.2 - Princípios Artísticos

Cada forma de arte tem o seu meio de expressão próprio e esse meio é especialmente apto para um tipo de comunicação específico. A arte na *Web* pode ser vista na forma de uma combinação de textos, imagens, cores, alinhamentos, posicionamentos e movimentos que não sejam baseados unicamente em técnicas de uso de ferramentas. A arte é uma forma de inquérito que descobre, cria e alarga o conhecimento quando mostrada como um produto da cognição.

Um produto da cognição quando levado a extremos pode fazer com que o artista seja o único entendedor do seu produto, restando aos contempladores apenas levantar possibilidades a respeito da intenção do artista. Não é bem esta forma de arte que se emprega na *Web*, onde a arte tem espaço nas imagens usadas como forma de conhecimento complementar à escrita e a formas estéticas atraentes ao usuário (caracterizações dos atributos).

A educação artística fornece capacidades de leitura, de codificação e decodificação de mensagens visuais. Também tem como finalidade evidenciar os fatos para a observação e compreensão das representações visuais das diferentes culturas. A abordagem da educação artística pode organizar-se segundo os campos da estética, da crítica, da história e da produção. A contextualização, a análise crítica, a reflexão estética e o reconhecimento da pluralidade cultural são estratégias essenciais segundo o conceito de "cultura visual" para o entendimento de uma mensagem.

A cultura visual representa um "treino" da visão para entender e conseqüentemente desenvolver formas de trabalho que o conhecimento somente técnico não permitiria. Um exemplo disso pode ser visto quando diferentes profissionais fazem uso da mesma

ferramenta, para produzir um mesmo produto (interface, logotipo, imagem complementar a um texto etc.) e os resultados são diferentes, conforme suas habilidades. Cada categoria de profissionais explora um sentido conforme a arte com a qual trabalha como, por exemplo, um músico precisa treinar a audição, um cozinheiro precisa treinar o paladar e um *Web designer* precisa treinar a visão. A visão é o sentido mais utilizado pela maioria dos seres humanos e conseqüentemente o que proporciona as maiores condições de imaginação e de criação.

Howard Gardner (Gardner, 1997), autor da teoria das inteligências múltiplas mostra que a "visão espacial" está relacionada ou é proveniente ao sentido da "visão" e que o uso dessa forma de inteligência é que faz com que o indivíduo desenvolva uma observação minuciosa das coisas, das formas, das imagens mentais etc.

Habilidades como esta são essenciais a um *Web designer*, assim como o são a um artista. No entanto, um *Web designer* com conhecimentos artísticos está apto a desenvolver trabalhos qualificados, com estética atraente, e fáceis de serem entendidos pelo usuário. Por outro lado, se um artista, que tenha pouco ou nenhum conhecimento de *Web design*, trabalhar no desenvolvimento de *Web sites*, o resultado tende a ser desastroso, pois não se pode prever o perfil do usuário que entenderá seu trabalho.

Gardner defende a existência de diferentes modalidades de desenvolvimento enraizadas em diferentes modalidades de inteligência. Considera que os seres humanos evoluíram através dos milênios a ponto de serem capazes de realizarem pelo menos sete formas diferentes de conhecimento ou de "processamento da informação". Estas incluem formas de inteligência que lidam com linguagem, lógica e matemática, musical e rítmica, informação visual espacial, informação corporal-cinestésica, interpessoal e intrapessoal. Todos os seres humanos normais possuem alguma capacidade em cada uma destas esferas ou perfis de inteligências:

- **Lingüística**: relacionada às linguagens faladas, significados e relações entre palavras. Domina a maioria dos sistemas educacionais ocidentais.
- **Lógica / matemática**: relacionada ao pensamento dedutivo e raciocínio, números, pensamentos abstratos, precisão e estrutura lógica.
- **Rítmica / musical**: relacionada ao reconhecimento de padrões de tons, incluindo sons ambientais, sensibilidade ao ritmo, chave, batidas, o poder emocional e a organização complexa da música.

- **Visual / espacial**: relacionada ao sentido da visão, observação minuciosa, metáfora, pensamento visual, imagens mentais e habilidade de formar figuras tridimensionais na mente.
- **Corporal / sinestésico**: relacionada ao movimento físico, ao controle do corpo e de objetos, ao tempo e ao conhecimento/sabedoria do corpo.
- **Interpessoal**: relacionada a relacionamentos entre pessoas e à comunicação, sensibilidade para com os outros, habilidade para ler intenções e desejos alheios, habilidade para influenciar os outros.
- **Intrapessoal**: relacionada ao auto conhecimento, estados interiores do ser, auto-reflexão, meta cognição e consciência de valores temporais e espirituais, propósitos e sentimentos.

Gardner não reconhece a existência de uma inteligência artística, inteligência estética ou inteligência perceptual, mas que os sistemas simbólicos são mobilizados para fins artísticos quando os indivíduos exploram esses sistemas de determinadas maneiras e para determinados fins. Não existe uma inteligência artística separada, mas o direcionamento de cada uma das formas de inteligência, mencionadas anteriormente, para fins artísticos. O que quer dizer que os símbolos vinculados nessa forma de conhecimento podem ser (mas não obrigatoriamente) ordenados esteticamente (Gardner, 1997).

O olhar é o sentido artístico por excelência, pois é pelo olhar que mais se adquire e acumulam-se conhecimentos que se faz da leitura nos mais diversos recursos e dos mais diversos tipos de linguagens (Gallo, 1997).

Do ponto de vista didático pode-se separar e mostrar algumas funções da arte como, por exemplo, a função pragmática ou utilitária da arte e a naturalista.

- **Função pragmática ou utilitária da arte**: dentro desta visão, a arte serve ou é útil para se alcançar um fim não-artístico, isto é, ela não é valorizada por si mesma, mas como um meio de alcançar uma outra finalidade. Na Idade Média, por exemplo, esse tipo de arte serviu para ensinar os principais preceitos da religião católica relatando histórias bíblicas para a população dos feudos, onde a maioria era analfabeta.

- **Função naturalista da arte**: a função naturalista refere-se aos interesses pelo conteúdo da obra, ou seja, pelo o que um trabalho retrata, em detrimento de sua forma ou modo de apresentação. O trabalho é encarado como um espelho, que reflete a realidade. Essa tendência caracterizou a arte ocidental ate meados do século XIX, quando surgiu a fotografia (Aranha, 2003).

É apenas do ponto de vista didático que se podem separar as funções da arte, pois elas podem se apresentar juntas dependendo da finalidade ou de como um expectador percebe uma arte.

3.1.2.3 - Princípios Éticos

Há questões que não representam somente um ponto de vista ou um modelo de *design*; que não tratam simplesmente da escolha de uma linguagem e/ou tecnologia em detrimento de outra, mas como fazer um uso melhor das tecnologias escolhidas.

Determinadas questões como estética, gostos, preferências, facilidades de leitura e navegação, não são problemas que a engenharia possa solucionar, mas são questões que precisam ser pensadas e analisadas em busca de levar aos usuários o melhor possível dentro dos limites de cada projeto (tempo, verbas, recursos em geral). Questões como estas são abordadas, neste trabalho, sob vários aspectos, inclusive sob aspectos filosóficos, como da ética, da estética e a arte.

Em interfaces *Web* podem-se considerar questões éticas em aspectos como as imagens, textos, o conteúdo dos textos escritos, a forma de organização do conteúdo e do uso das tecnologias, na busca de levar até os usuários facilidade de uso do sistema e clareza nas informações.

Sobre a ética das imagens, segundo, Peixoto, in: (Novaes, 2003), "*a ética das imagens é dar tempo e lugar para as coisas. Aquilo que elas precisam para ser. Imagens que procurem respeitar o tempo e o espaço para que as coisas se cristalizem diante dos nossos olhos. Ética é saber* ***atentar*** *para o tempo e o lugar das coisas (...) Imagens que procurem olhar o mundo nos olhos, que tentem deixar as coisas no olhar. Perceber aqui o que faz as coisas falarem: a sua luz, o seu rio subterrâneo. Essa atitude - esse respeito pelas coisas - é ético. Olhar o mundo como uma paisagem, algo dotado de luz, de uma capacidade de nos responder ao olhar. Não se trata de procurar cenas naturais, mas de um modo de ver. Ver rostos e cidades como paisagens: uma ética do olhar*" (Novaes, 2003).

Sobre o uso das tecnologias, faz-se feita uma analogia com o atomismo, segundo, Pessanha, in: (Novaes, 2003). "*A ética exige mais que um mecanicismo. Exige uma racionalidade flexível para dentro dela caber o humano com seus projetos; o humano com seus ideais; o humano com suas metas, ou seja, o atomismo explica todas as coisas, só não explica apenas que o homem seja uma coisa que tenha que ficar inerte diante da fatalidade do jogo mecânico. O homem é racional, mas a racionalidade do mundo precisa dar conta também racionalmente da ação do homem contrária à fatalidade das coisas. O homem é livre e é isso que é a grandeza de sua liberdade, porque apesar da fatalidade das coisas, do mecanismo do mundo, ele abre uma brecha para seus projetos, ele é o inventor do* ***dever ser****, acima da fatalidade das coisas que são. Ele introduz uma dimensão que é a dimensão do dever ser, do seu projeto de vida, da sua meta... Ele não é um ser passivo diante do mundo; ele não é apenas um reflexo das circunstâncias, ele não está apenas à mercê das contingências. Ele é alguém capaz de, diante das contingências, dar respostas diferenciadas e por isso mesmo ele pode estabelecer o seu itinerário; a sua meta; a sua rota* (Novaes, 2003).

Alguns autores, algumas categorias profissionais e alguns segmentos sociais fazem uso da ética como sendo uma espécie de "obrigação" ou de verdades absolutas que tenham que ser seguidas. Porém, neste trabalho, a ética é abordada como uma cultura a partir da qual pode-se levar até os usuários interfaces com uma funcionalidade melhor, devido ao uso diferenciado de tecnologias e de um *design* baseado em observações de dificuldade (cultura, prática de uso, recursos de *software*, *hardware*, banda etc) de alguns usuários. Propor em um processo de desenvolvimento *Web*, um padrão de *design* poderia representar uma falta de percepção e de reconhecimento da "capacidade de criação". Assim, na terceira fase do processo de desenvolvimento apresentado neste trabalho, são sugeridas algumas questões que "podem" auxiliar um profissional na escolha das tecnologias, nas opções de *design* e no uso geral de textos e imagens.

Sobre a ética vista como um tipo de obrigação, segundo, Ribeiro, "*os códigos de ética, na verdade, são leis disfarçadas, leis light, promulgadas por quem não tem poder para legislar (por exemplo, uma empresa, uma associação profissional) (...) Os códigos de ética já se mostram mais códigos do que éticos. Porque há um velho conflito entre a ética e o código, ou seja, entre a ética e a lei. Para a lei, basta que sejam obedecidas. Mas, para a ética, isso não quer dizer nada. Se eu cumprir a lei por medo das conseqüências, meu ato não tem nada de ético. (...) O grande problema dos códigos de ética é o seguinte: eles podem levar as*

pessoas a pensar que são éticas a baixo custo. Bastará obedecermos a suas disposições e, pronto, seremos éticos. Numa sociedade que questiona inúmeras coisas, que está atravessada por dúvidas (o que é muito positivo, porque desperta a pergunta ética), um código pode ser a resposta fácil para um problema complexo. Pode calar a pergunta pela decência, em vez de dar-lhe o devido valor. Provavelmente a sociedade vai continuar gerando códigos de ética, e o resultado básico é bom, sobretudo se eles decorrerem de uma ampla discussão social, porque assim se envolve todo um grupo ou categoria profissional. Mas devemos sempre deixar claro que nenhum código de ética vai fazer uma pessoa ética" (Ribeiro, 2004).

A questão da ética das interfaces *Web* será mostrada no capítulo 5, pois o assunto é parte integrante do trabalho proposto. Não que se pretenda criar um "código de ética" para as interfaces *Web*, mas ao contrário, mostrar algumas questões podem levar mais qualidade aos usuários, dando tempo e espaço à arte e à tecnologia. Para propõe-se alguns princípios culturais como estética, arte é ética em busca de melhores *designs*, respeitando a capacidade de criação que gera *designs* atraentes sem comprometer o bom funcionamento navegacional.

Questões Legais não serão abordadas neste trabalho.

3.2 - Práticas Recomendadas para a Internet (IEEE 2001)

O IEEE-2001 (Instituto de Engenheiros Elétricos e Eletrônicos) é um padrão voluntário conhecido como "Práticas recomendadas para a Internet - Engenharia *Web*, gerenciamento de *Web sites* e padrões de ciclos de vida de *Web sites*". O padrão conhecido como IEEE 2001, sobre benefícios e normas de procedimentos para desenvolvimentos de *Web sites*, incluiu os tipos de informação que devem ser destacadas em todos os *Web sites*, tal como informações sobre o criador do *Web site*; URLs e datas de *updates*.

Usuários acessam páginas em busca de dados confiáveis e atualizados. Por isso um *Web site* deve trazer a identificação do que é, e a quem pertence; o propósito do *Web site*; se tem fins lucrativos ou é somente informativo; quem são as pessoas responsáveis pelas divisões e a forma de contatá-las; a política de privacidade, como por exemplo, se o *Web site* tem *cookies* que armazenam dados do usuário; a propriedade intelectual das informações disponíveis; se as informações estarão permanentemente no *Web site* ou se são temporárias e quando são temporárias, qual é a data de expiração, de atualização e a última modificação.

As páginas *Web* devem ser feitas considerando o acesso por usuários diversificados. O IEEE 2001 (Internet, melhores práticas de EW) sugere *Web sites* que procurem uma conformidade para implementar formas de procedimentos de acesso. Os métodos de melhoramentos de acesso para pessoas leigas, melhoram o acesso para todas as demais. Usuários mais freqüentes *on-line* agem de acordo com as normas que representam credibilidade aos *Web sites*.

Exemplos que facilitam o sistema de acesso e navegação:

- Reprodução do conteúdo para um sistema facilitador de áudio como, por exemplo, onde se possam escolher traduções para outros idiomas e informações do conteúdo em mecanismos de busca;
- Internacionalização de informações de contato como números de telefones contendo o código do país, da cidade e o fuso horário. Onde houver produtos que têm preços, deve ter tabelas de conversão para diversas moedas, como por exemplo, $US, $pesos, $Hong Kong, R$reais etc.;
- Mecanismos de busca eficientes devem ser marcados por *meta tags* que identifiquem conforme a data da última modificação, da expiração, e o endereço da *home page*;
- Modelos eficientes e abrangentes de divulgação, reconhecimento e aprovação, devem fazer uso de normas de processos e ferramentas de desenvolvimento de *Web sites* que correspondam as normas de especificações (Isaak, 2001).

As práticas padronizadas pelo IEEE representam uma conscientização de como melhorar a compreensão e a forma de navegação de um usuário dentro de uma página. Além dos itens mostrados acima, o padrão de melhores práticas de EW, tem muitos outros parágrafos que se referem à usabilidade da *Web*.

3.3 - W3C

O W3C (*World Wide Web Consortium*) é a principal organização promotora e padronizadora da *Web*, mundialmente. A própria especificação do HTML, XML, e outras linguagens, são desenvolvidas pelo W3C. O W3C foi fundado em Outubro de 1994 para levar a *World Wide Web* a atingir seu potencial máximo através do desenvolvimento de protocolos comuns que promovam sua evolução e garantam sua interoperabilidade. Atualmente a W3C tem mais de

450 Membros e um quadro de aproximadamente 70 pessoas em tempo integral globalmente, que contribuem para o desenvolvimento de especificações de W3C e *software*.

A missão do W3C é levar a *Web* ao seu potencial máximo, que se consegue através do desenvolvimento de tecnologias (especificações, diretrizes, *software* e ferramentas) criar um fórum para informação, comércio, inspiração, pensamento independente e compreensão coletiva. Este sumário com sete pontos apresenta os objetivos do W3C e os princípios operativos.

- **Acesso Universal**: o W3C define a *Web* como o universo de informação acessável por rede (disponível através de seu computador, telefone, televisão). Atualmente este universo beneficia a sociedade através da oferta de novas formas de comunicação entre humanos e oportunidades de compartilhamento de conhecimento. Um dos primeiros objetivos do W3C é tornar estes benefícios universais para todas as pessoas, independentemente de *hardware*, *software*, infra-estrutura de rede, linguagem nativa, cultura, localização geográfica ou capacidades mentais ou físicas. As atividades do W3C são atividades de internacionalização, atividade independente de aparelhos, atividade de *browser* por voz, e iniciativa de acesso à Internet (todos com versão em inglês), ilustram o compromisso para um acesso universal.

- ***Web* Semântica**: as pessoas atualmente compartilham seu conhecimento na *Web* em uma linguagem destinada a outras pessoas. Na *Web* Semântica (semântica, significa relacionado a significado) se expressa de modo a que os computadores possam interpretar e fazer as trocas com as aplicações. Assim, resolvem-se problemas e propiciam-se formas de ajuda para que se encontre mais rapidamente o que procura, como por exemplo, informação médica, comentários sobre um filme, uma ordem de compra de um livro etc. As linguagens do W3C, RDF, XML, esquema XML, e assinaturas XML são os elementos fundamentais da *Web* Semântica.

- **Confiança**: a *Web* é um meio de colaboração e não apenas uma revista de leitura. Na realidade, o primeiro navegador para a *Web* era também um editor, apesar de a maioria das pessoas imaginarem os navegadores com uma função principal de visualização e não interação. Para promover um ambiente mais colaborativo, torna-se necessária a existência de uma rede de confiança que garanta confidencialidade,

passe confiança e torne possível às pessoas tomar responsabilidades por (ou sejam responsáveis por) aquilo que está publicado na Rede. Estes objetivos orientam muito do trabalho no W3C's sobre assinaturas XML, mecanismos de anotação, autoridades de grupo, versões etc.

- **Interoperabilidade**: há vinte anos as pessoas compravam *software* que funcionava apenas com algum outro *software* desde que fosse do mesmo vendedor. Atualmente as pessoas têm mais liberdade de escolha e corretamente esperam que os componentes de *software* sejam intercambiáveis. Também se espera que seja possível visualizar conteúdo existente na rede com o *software* de sua preferência (navegador de PC gráfico, sintetizador de voz, navegador braille, telefone do carro etc.). A W3C, uma organização neutra a vendedores, promove interoperabilidade através do desenvolvimento e promoção de linguagens de computador abertas (não proprietárias) e protocolos que evitam uma fragmentação do mercado que existia no passado. Estes pontos são conseguidos através de consenso na indústria e encorajamento de uma discussão em fórum aberto.

- **Evolução**: o W3C visa a excelência técnica, porém está ciente que o que conhecemos e necessitamos atualmente pode não ser suficiente para a solução de problemas no futuro. Assim, busca o desenvolvimento de uma rede que possa facilmente evoluir para uma rede ainda melhor, sem quebra de funcionalidades anteriores. Os princípios da simplicidade, modularidade, compatibilidade e extensibilidade, orientam todo o desenvolvimento W3C.

- **Descentralização**: a descentralização é um princípio de sistemas distribuídos, incluindo as próprias sociedades. Em um sistema centralizado toda a mensagem ou ação tem que passar por uma autoridade central, originando gargalos sempre que o tráfego aumenta. Em conceito, limita-se então a quantidade de pontos centrais na rede para reduzir a vulnerabilidade na Rede, como um todo.

- **Melhor multimídia**: quem não gostaria de mais interatividade e uma melhor mídia na rede, incluindo-se aqui imagens que podem alterar seu tamanho, som de qualidade, vídeo, efeitos tri-dimensionais e animação? O processo de consenso no W3C não limita a criatividade de fornecimento de conteúdo ou significa visualizações de conteúdo “pobres”. Através de seus membros o W3C escuta os

usuários finais e trabalha com o objetivo de fornecer bases sólidas para o desenvolvimento de uma Melhor Multimídia através de linguagens como a linguagem *Scalable Vector Graphics* (SVG) e a *Synchronized Multimedia Integration Language* (SMIL) (W3C, 2003).

3.4 - UML

Durante o período de 1989 a 1994, a quantidade de métodos orientados a objetos aumentou consideravelmente, em pouco tempo. Com isso, muitos usuários desses métodos tiveram dificuldades para encontrar uma linguagem de modelagem capaz de atender as suas necessidades e isso acabou gerando a chamada "guerra dos métodos". Com os problemas decorrentes no desenvolvimento de *softwares*, surgiu a necessidade de uma linguagem unificada para que os desenvolvedores de *softwares* pudessem falar uma linguagem comum (Booch, 2000).

Para apontar uma solução à "guerra dos métodos", os autores Grady Booch, Ivar Jacobson e James Rumbaugh se uniram e unificaram seus métodos, criando a *Unified Modeling Language* (UML). A UML trouxe uma certa estabilidade ao mercado de desenvolvimento de *software* permitindo que os projetos tivessem como base uma linguagem de modelagem mais. Quando a UML foi lançada, muitos desenvolvedores da área da orientação a objetos foram beneficiados já que essa padronização proposta pela UML trouxe o apoio que eles esperaram (Booch, 2000).

Ao longo de 1996, a UML foi aceita pela comunidade de Engenharia de *Software* em geral, que atualmente considera a UML uma grande aliada, pois permitiu que os desenvolvedores de *softwares* pudessem falar uma linguagem comum.

Utilizar a UML em projetos *Web* não significa explorar todos os significados de classes, objetos, relacionamentos, fluxos, mensagens e outras entidades comuns da orientação a objetos, mas fazer uso dos recursos pertinentes a UML em busca de melhores projetos de *Web site*. As melhorias nos projetos podem ser vistas no uso de diagramas como, por exemplo, diagramas de seqüência, diagramas de caso de uso, diagramas de classes e demais diagramas que possam ser utilizados na modelagem de projetos de *Web sites*.

O objetivo de buscar a modelagem de interfaces *Web* através dos recursos da UML é fazer com que haja uma linguagem comum entre desenvolvedores e administradores de *Web sites* no projeto, implementação e administração. Através de projetos objetivos e estruturas

padronizadas é possível uma aproximação maior do ideal de fazer com que os *Web sites* possam ser sistemas "orientados ao usuário" (Booch, 2002).

A UML é uma tentativa de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível (Booch, 2000).

Para entender melhor um sistema a ser desenvolvido a UML faz uso de modelagens e padrões de contexto, como os descritos a seguir.

3.4.1 - O Uso da Modelagem para Entender Melhor o Sistema

Um modelo ou um processo pode representar uma simplificação da realidade. Os modelos fornecem uma cópia do projeto de um sistema e podem abranger planos detalhados ou planos gerais com uma visão panorâmica do sistema considerado. Um bom modelo inclui aqueles componentes que têm ampla repercussão e omite os componentes menores que não são relevantes em determinado nível de abstração. Todos os sistemas podem ser descritos sob diferentes aspectos, com a utilização de modelos distintos, sendo cada um deles uma abstração semanticamente específica do sistema.

Os modelos podem ser **estruturais**, dando ênfase na organização do sistema; **comportamentais**, dando ênfase à dinâmica do sistema (Booch, 2002), (Mesquita, 2003).

3.4.2 - Padrões

A noção de padrões tem as suas origens no trabalho do Engenheiro civil Christopher Alexander. Durante 10 anos, Alexander recolheu e documentou soluções genéricas para problemas recorrentes no domínio da arquitetura. O objetivo inicial foi o de habilitar não-especialistas a projetar as suas próprias casas e comunidades. Alexander mostrou que "*Cada modelo descreve um problema que ocorre inúmeras vezes em nosso meio e então descreve a essência da solução daquele problema de maneira que podemos usar esta solução milhões de vezes, sem que isto seja feito duas vezes da mesma maneira*" (Mesquita, 2003).

Apesar de Alexander estar se referindo a padrões da Arquitetura, o que ele disse também é verdadeiro para os padrões de projeto orientados a objeto. Na Orientação a Objetos (OO)

fala-se em objetos e interfaces ao invés de paredes e portas, mas o núcleo dos dois tipos de padrão é a solução de um problema em um contexto determinado.

Os Formatos dos Padrões de Alexander podem ser entendidos como:

- **Nome**: uma frase ou nome descritivo identificativo do padrão. Exemplo, fotografias, desenhos e descrições que ilustram a aplicação dos padrões.
- **Contexto**: as circunstâncias nas quais o padrão pode ser aplicado. Explica a razão da existência do padrão e a sua generalidade.
- **Problema**: descreve as forças relevantes e restrições e como interagem entre si.
- **Solução**: relações e regras dinâmicas que descrevem como construir artefatos em concordância com o padrão. São referidos padrões similares e variantes, bem como padrões de níveis superior e inferior relevantes para a concepção da solução proposta (Mesquita, 2003).

3.5 - Engenharia de *Web sites*

A Engenharia *Web* (EW) pode ser definida como "Aplicação de um enfoque sistemático, disciplinado e quantificável para o desenvolvimento e evolução de aplicações *Web*, com alta qualidade e a um custo efetivo" (Graef, 2000).

A EW lida com diversas áreas: hipermídia, banco de dados, interação homem-computador, artes, comunicação, lingüística computacional, gerência, apresentação gráfica, computação gráfica. Tudo isso resulta em uma combinação entre:

- Publicação impressa e desenvolvimento de *software*;
- *Marketing* e computação;
- Comunicação interna e relações externas (Graef, 2004)
- Arte e tecnologia (Domingues, 2003).

A Engenharia *Web* não é idêntica a Engenharia de *Software*: métodos, técnicas e ferramentas tradicionais de desenvolvimento de *software* não são sempre adequados para o projeto de *Web sites*. Os *Web sites* são produtos de *software* e apresentam requisitos próprios, como a necessidade de gerenciar um grande volume de informações, combinar a navegação controlada pelo leitor com a própria natureza das informações multimídia, atualizações constantes, entre outros fatores (Zafiris, 2001).

Surge assim um novo paradigma de engenharia denominado Engenharia de *Web sites* ou Engenharia *Web*, que define um processo para construir aplicações hipermídia na Internet com qualidade, com base em critérios tais como: necessidades dos usuários, conteúdo de alta qualidade, atualizações constantes (manutenção), tempo de *download* mínimo, usabilidade e cultura corporativa centralizada na rede (Zafiris, 2001).

3.5.1 - Problemas no Desenvolvimento de *Web Sites*

- ***Web sites* mal definidos e mal projetados**: resultam em *Web sites* "emendados", com interfaces inadequadas, difíceis de se entender e navegar.
- **Métodos de implementação inadequados e obsoletos**: um *Web site* não deve ser desenvolvido somente com tecnologia de ultima geração (sem que seja devidamente testada), uma vez que não é possível conhecer os recursos de requisição (*software*, *hardware* e banda) de cada usuário. Nem por isso deve-se usar uma versão muito antiga de uma linguagem ou tecnologia, privando os usuários de interfaces mais intuitivas e dinâmicas.
- **Falta de documentação e dificuldades de implementação e manutenção**: A EW faz uso dos recursos da ES, da UML e da engenharia da usabilidade, mas ainda encontra dificuldades para elaborar projetos, documentação, implementação, *design* e manutenção de *Web sites*. Isso significa que ainda há dificuldades no uso uma linguagem comum entre os profissionais da *Web*. A necessidade de organizar o material de maneira adequada requer uma abordagem sistemática. A falta de diretrizes, modelos e métodos para a criação tornam o processo difícil e desafiante.
- **Visão da atividade como arte ou como engenharia**: a concepção de que o "desenvolvimento de *Web sites* é uma arte" está errada. Existe um importante lado artístico/*design*, mas os desenvolvedores precisam também de disciplina e de um modelo de processo.
- **Inexperiência e formação inadequada dos desenvolvedores**: com o surgimento de *softwares* geradores de HTML e a facilidade do uso desta linguagem, muitos profissionais de outras áreas puderam desenvolver seu próprio *Web site*, colocando o conteúdo disponível *on-line*, sem critérios, estimativas de acesso ou controle da

complexidade. Isso levou à idéia de que trabalhar com *Web sites* é "fácil": basta enviar os arquivos desenvolvidos para uma máquina servidora.

- **Custos mal dimensionados**: o trabalho de profissionais qualificados, como em qualquer outra área, tem um custo considerável. Além disso deve-se considerar, os custos de aquisição e manutenção para uso de um domínio, custo de hospedagem, tecnologias de desenvolvimento e manutenção. Sem um projeto objetivo e bem definido, fica difícil mensurar os custos decorrentes (Zafiris, 2001 (Kirda, 2001).

3.5.2 - Diferenças entre a Engenharia de *Software* e a Engenharia *Web*

- **Diferentes propósitos:** aplicações convencionais projetadas tipicamente pela ES oferecem serviços e soluções enquanto *Web sites* oferecem conteúdos que são informações e/ou serviços, representando assim, um diferente meio de comunicação.
- **Apresentação x funcionalidade**: aplicações convencionais dão ênfase à funcionalidade e aplicabilidade, enquanto *Web sites* dão ênfase à apresentação, aparência, navegação e a outras qualidades estéticas.
- **Tradição x experiência**: existe mais experiência no desenvolvimento de *software* convencional, possibilitando planejamento e gerência mais realísticos em relação a processos de *Web sites*.
- **Público alvo x audiência (Classe de usuários)**: aplicação convencional tem classe de usuários mais bem definida, com ênfase na aplicabilidade e funcionalidade. Já os *Web sites* têm um público diversificado com ênfase na navegação e qualidades estéticas.
- **Maturidade da Tecnologia**: aplicações convencionais têm tecnologias mais estáveis e excelentes ferramentas disponíveis, enquanto a EW tem tecnologias em constante evolução, diversas ferramentas disponíveis para implementação, mas existem poucas para análise, projeto, documentação e outras atividades da EW (Kirda, 2001).

3.5.3 - Principais Atividades

As principais atividades de desenvolvimento de *Web sites* são a definição do problema, motivação, propósitos, estimativas de acesso, planejamento e gerência do projeto, estudo de viabilidades, análise e seleção de requisitos, *design*, estrutura organizacional, estrutura das interfaces, sistema de navegação, conteúdo, funcionalidade, implementação, testes e manutenção.

3.5.4 - Principais Tipos de Requisitos para *Web Sites*

Para o desenvolvimento de *Web sites* são necessários alguns requisitos que servem de base para o planejamento do conteúdo, da estrutura das interfaces, do sistema navegacional e do *design*.

- **Requisitos de conteúdo**: quais informações o *Web site* deve conter? Visão e modelagem das informações do *Web site* devem determinar como as páginas serão organizadas. Para a interface da página deve-se planejar a organização, interação e apresentação que irão determinar os aspectos estéticos e visuais. Deve-se definir a diagramação das páginas, o uso de imagens e ícones, idioma, cores, tipo de fontes, plano de fundo.
- **Requisitos funcionais**: qual a tecnologia necessária? Quais serviços o *Web site* deve oferecer? Qual a arquitetura dos programas, projeto de banco de dados, plataforma? Que tipos de testes serão realizados?
- **Requisitos de interação e navegação**: visão de navegação como uma forma de entendimento de como o usuário vai utilizar/interagir/navegar no *Web site*. É necessário determinar como os menus, âncoras, botões e ícones podem ser utilizados.
- **Requisitos de projeto e de desenvolvimento**: os requisitos de desenvolvimento são referentes ao número e à qualificação das pessoas envolvidas; os prazos revistos para a entrega de etapas específicas e para a conclusão e os custos previstos para o projeto e para a implementação. *Web sites* devem prever as tecnologias que serão utilizadas, prever a compatibilidade com os recursos do servidor para que seja possível uma previsão funcional, de cronograma e orçamentária. Exemplo:

- Imagens: Adobe Illustrator, Photoshop, Macromedia Freehand, Fireworks.

- Edição/Codificação: Dreamweaver, Frontpage, Adobe GoLive.

- Diagramação: Adobe Pagemaker, Microsoft Visio, Rational Rose.

- Animações: Macromedia Flash.

- Linguagens: HTML, DHTML, JavaScript, CGI, ASP, PHP, Java Servlets, XML ou outras (Kirda, 2001).

Princípios, normas e padrões representam formas de cultura que auxiliam os profissionais a tomar decisões e a fazer melhores escolhas dentro dos limites de cada projeto. No entanto, a *Web* é dinâmica e conta com desenvolvedores e usuários com os mais diversos níveis de instrução, exigência, capacidade de discernimento e avaliação. Se um *Web designer* se ativer somente a normas e padrões e tentar colocá-los como verdades absolutas que se atribuem a todos os domínios e a todas as situações, correrá o risco de tornar seus trabalhos obsoletos e previsíveis demais diante das exigências do mercado. Uma das possíveis ponderações para o uso de princípios, normas e padrões é que sejam utilizados com atenção e cuidado para que um profissional não anule a sua capacidade criativa e nem desenvolva trabalhos com excessos de formalidades.

CAPÍTULO 4
ESTRUTURAS E MODELOS DE *WEB SITES*

Nesta Seção explicam-se as estruturas e os modelos de projetos mais utilizados em *Web sites*.

4.1 - Estruturas de *Web sites*

Para que uma interface seja elaborada é necessária a adoção de um tipo de estrutura para a mesma. A estrutura de uma interface *Web* é definida pelo esboço onde serão colocados os conteúdos (Chase, 2000). Existem vários tipos de estruturas, os quais são escolhidos pela facilidade no manuseio dos arquivos, pela necessidade do conteúdo ou pelos recursos disponíveis no servidor.

Ao iniciar um projeto de *Web site* é necessário definir o tipo de estrutura, pois os arquivos terão que ser desenvolvidos de acordo com as características da estrutura, como por exemplo, o uso do parâmetro *target* em *links*, que define em que tipo de janela uma página será exibida.

Destacam-se a seguir, os tipos mais freqüentes de estrutura encontradas em *Web sites*.

4.1.1 - Página Única com *Links* para os Tópicos

Essa é a forma mais simples de se expor conteúdos *Web*. A composição se dá com uma única página onde há *links* que apontam do início da página para um tópico específico, do final para o início e de qualquer ponto da página para onde houver um *link* (esta estrutura é bastante utilizada em *Frequently Asked Questions* - FAQs). A estrutura de página única com *links* para os tópicos é feita com em uma única página onde os valores atribuídos aos parâmetros *target* e *name* das *tags*, "<A HREF>" e "<A NAME>" respectivamente, é que determinam a forma de navegação. O limite deste tipo de estrutura encontra-se no tamanho das páginas geradas quando o conteúdo é extenso. Quando há mais de um assunto a ser exposto e quando há necessidade de ilustrações feitas com imagens, esse tipo de estrutura não é o mais adequado.

4.1.2 - *Frames*

Os *frames* são estruturas vazias, que somente dividem a tela. Nos espaços vazios são inseridos arquivos com conteúdos. Normalmente um arquivo fica fixo, com a barra de navegação fazendo a chamada das páginas que compõem o *Web site*. A estrutura de *frames* requer um arquivo com o menu e os demais com o conteúdo conforme indicam os *links* e o parâmetros *target* é que fará com que um arquivo seja exibido em uma janela cujo nome foi atribuído no arquivo do *frame* que faz a divisão da tela. Este sistema é bastante utilizado devido à praticidade no desenvolvimento, porém para o usuário, apresentam problemas funcionais e estéticos, como os explicados a seguir.

- **Funcionais**: pode causar acoplamentos de janelas, isto é, fazer com que um arquivo seja aberto dentro do espaço do outro, reduzindo o espaço para a visualização do conteúdo. Pode ainda, dificultar o acesso a tópicos específicos devido ao endereço destes ser camuflado no *location* do navegador e não funcionar em janelas *pop-up* na maioria dos navegadores.
- **Estéticos**: apresentam problemas estéticos como a divisão da tela que dependendo do conteúdo geram barras de rolagem em pontos estratégicos; não há como padronizar as interfaces, pois eliminam a barra de navegação quando se chama um *link* para um tópico isolado (Baker, 2001), (Roselli, 1999).

4.1.3 - Páginas Divididas com Tabelas

É um sistema onde se elabora um arquivo com uma tabela que contém uma barra de navegação em uma das linhas ou colunas e o conteúdo em outra divisão (linha/coluna). Elabora-se um *script* contendo o menu e este *script* é copiado e colado em todas as páginas, fazendo com que o usuário veja as páginas escolhidas, acompanhadas da barra de navegação. A estrutura de divisão com tabelas requer que todo o conteúdo esteja exposto em cada página e o valor do parâmetros *target* é que fará a substituição das páginas que é a forma de navegação.

Este tipo de estrutura revela suas limitações em relação ao número de arquivos e diretórios que compõem um *Web site*. A necessidade de copiar e colar o *script* em todas as páginas não representa grandes problemas para um *Web site* com poucos arquivos, mas considerando um grande número, este sistema torna-se impraticável.

4.2 - Modelos de *Web Sites*

Nesta Seção faz-se um levantamento de alguns modelos de projetos, desenvolvimento e administração de *Web sites*.

No desenvolvimento de *software*, nos modelos mostrados na fundamentação deste trabalho, bem como, no processo proposto, usa-se muito o termo "estereótipo" para fazer referência a ilustrações necessárias ao entendimento do que ocorre em diversas fases e níveis de desenvolvimento de *softwares*. Este termo pode ser entendido como:

- Etimologicamente (Simões, 1985), o termo estereótipo designa uma placa metálica de caracteres fixos destinada à impressão em serie. Trata-se de um termo que embora provindo do vocabulário tipográfico, adquiriu uma conotação psicossocial, remetendo para "uma matriz de opiniões, sentimentos, atitudes e reações dos membros de um grupo, com as características, rigidez e homogeneidade" (Lima, 1986).
- De um ponto de vista mais estritamente cognitivo (Codol, 1989), a estereotipia identifica-se com a prototipia, tratando-se de uma "operação que consiste em atribuir a objetos de uma categoria todos os traços que se supõe caracterizar o conjunto dos objetos dessa categoria" (Lima, 1986).

4.2.1 - HDM

O *Hypertext Design Model* (HDM) (Garzotto, et all, 1993) prescreve um esquema de aplicação que descreve as classes abrangentes de informações dos elementos quanto às características comuns apresentadas, à organização interna da estrutura e a composição das interconexões mútuas. Portanto, este esquema captura a semântica e as regularidades estruturais na representação de estruturas para uma determinada classe de aplicações. Uma vez que um esquema seja especificado, o modelo também permite definir uma aplicação particular, provendo primitivas para descrever o exemplo do esquema, exemplos atuais de informação, classes e tipos de conexão.

O HDM é um dispositivo de modelagem que provê modos de descrever aplicações de hipertexto concisamente e de uma maneira independente do sistema existente. O esquema

ajuda o autor a formular os conceitos de uma determinada aplicação sem uma comunicação de linguagens entre *Web designers*, desenvolvedores e usuários.

A partir desta perspectiva, o modelo é o primeiro passo para o desenvolvimento e gerações de aplicações de modo semelhante as ferramentas CASE que são usadas na Engenharia de *Software*. A implementação de uma aplicação de HDM requer a definição de uma semântica de navegação, que especifique a dinâmica do comportamento e as propriedades de visualização da representação da estrutura HDM. Além disso o HDM especifica uma semântica padrão de navegação, mais apropriada para um sistema de classes.

O HDM pode ser usado como modelagem de projeto/dispositivo ou como uma implementação de projeto. Como uma modelagem, o HDM suporta um alto nível de especificações existentes ou de aplicações a serem desenvolvidas. Como uma implementação de projeto ele é uma base para ferramentas de *design* que suportam diretamente o desenvolvimento de aplicações. Em aplicações de hipertexto é a definição de um grande número de *links* que podem ser derivados do *design* conceitual no nível de descrições.

As entidades são naturalmente agrupadas em tipos de entidades, os quais correspondem a classes de objetos na vida real. Em aplicações de hipertexto, as entidades são freqüentemente um mesmo tópico que deve ser representado em diversas formas alternativas. Em aplicações multidimensionais, por exemplo, um mesmo tópico pode ser representado em diferentes idiomas. Ou ainda representar diversas maneiras de se ver a mesma situação, como por exemplo, um assunto visto em texto, vídeo, imagem som etc.

As primitivas do HDM são aplicações que consistem em estruturas feitas de grandes blocos de informação, chamadas entidades. Uma entidade denota um objeto físico ou conceitual de um domínio. Entidades são agrupadas pelo tipo da entidade. As semânticas de navegação têm o propósito de especificar como as estruturas de informação são visualizadas no cabeçalho e como se pode navegar através delas. O HDM provê um padrão relativamente simples que folheia a semântica que está embutida no cabeçalho. Diferentes semânticas de navegação, no entanto, podem ser definidas para descrever uma visualização mais sofisticada dos efeitos das aplicações especificadas. As primitivas HDM para a especificação de uma aplicação hipermídia são:

- **Entidade**: é a menor estrutura de informação autônoma que representa algum objeto do mundo real do domínio da aplicação. Sua existência independe da

existência de outros objetos. Entidades semelhantes podem ser agrupadas em um tipo de entidade.

- **Componentes**: as entidades são formadas por conjuntos de componentes não autônomos organizados hierarquicamente ao modo de árvores. Assim sendo, componentes possuem pai, irmãos e filhos e herdam as características da entidade da qual faz parte.

- **Perspectivas**: perspectivas representam maneiras distintas de apresentar um mesmo tópico atribuído a uma entidade. Todos os componentes desta entidade também herdam as perspectivas correspondentes.

- **Unidades**: as unidades são associadas a uma perspectiva específica. Caracteriza-se por um nome e um corpo, que é o local onde o conteúdo de uma aplicação hipermídia é preenchido.

- ***Links***: as ligações ou *links* podem capturar tanto as relações do domínio quanto padrões de navegação. Normalmente as relações do domínio não são consistentes com os padrões de navegação e vice-versa. Por isto, o HDM permite três tipos de ligações:

  - *Links* de perspectivas: ligações de perspectivas conectam unidades correspondentes de um mesmo componente. São fáceis de navegar porque mantém o tópico corrente inalterado.

  - *Links* de estruturas: ligam componentes pertencentes a uma mesma entidade. As ligações estruturais podem ser de vários tipos: *up* conecta um componente com o pai; *down-n* conecta um componente com o n-ésimo filho; *next-sibling* conecta um componente com o próximo filho do mesmo pai; *root* liga um componente à raiz da árvore. Permite que o usuário navegue entre trechos de informações pertencentes a mesma entidade, mantendo o mesmo contexto de informação.

  - ***Links* de aplicação e tipos de *links***: representam ligações complexas entre entidade e/ou componentes de uma aplicação hipermídia e que são dependentes do domínio. Ligações de aplicações podem ser agrupadas em tipos representadas por um nome, uma entidade fonte, uma entidade destino e um atributo de simetria. A

navegação por tipos de ligações implica na mudança brusca do contexto da informação.

Um correspondente de um tipo inverso de *link* pode ser especificado usando uma inversão de operador e suas instâncias podem ser automaticamente geradas. Em casos mais sofisticados pode-se especificar as regras de derivações tal como: "dando um tipo de *link* a aplicação", se a instância deste existir entre dois componentes de diferentes entidades, então uma instância do mesmo *link* é um tipo derivado entre as duas raízes que correspondem a entidade.

A Figura 4.1 mostra um exemplo de um componente de entidade ou tipo em um documento de "Leis", onde há "seção 1.1", em um contrato "X", que está sendo conectado pelo autor para o componente em uma entidade ou tipo de "Lei" - diga-se "Artigo 2" da lei "Y" por um *link* do tipo "justificado por". Então se assume que o autor queira representar um fato em diferentes níveis de detalhes onde todo o contrato "X", (é justificado) "*is-justified*" por toda a "Lei Y".

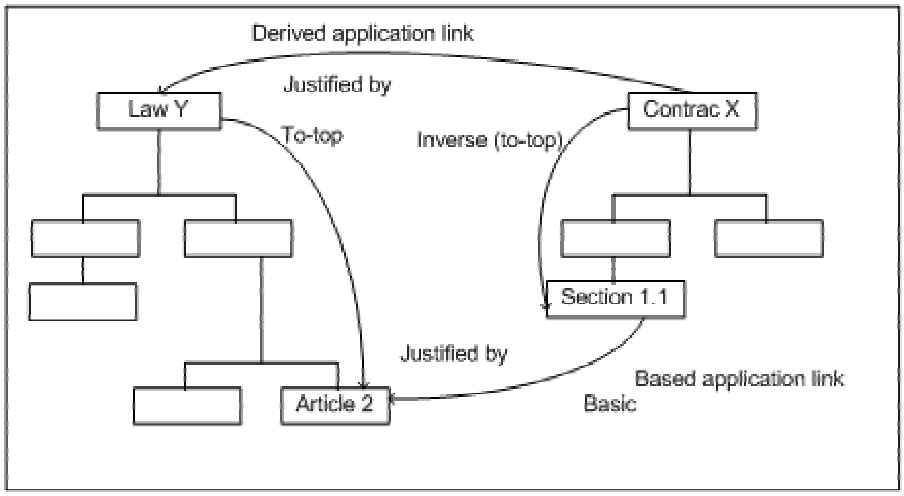


FIGURA 4.1 - Exemplo de *links* de uma aplicação derivada.
FONTE: Garzotto (1993).

O HDM mostra ainda normas de tradução de *links* de um estado abstrato para um concreto baseadas na idéia de se ter uma representação padrão para cada objeto abstrato (entidade ou componente). Isso corresponde a idéia de que o componente raiz na entidade, na perspectiva padrão, permanece para aquela entidade. Esta noção de aplicação de *links* de entidade-para-entidade traduz em *links* concretos entre seus padrões de representatividade. Cada *link* de aplicação componente-para-componente é traduzido em um tipo de *link* concreto conectando

cada unidade da fonte do componente para um padrão de representatividade do componente alvo (Garzotto, et all, 1993).

A Figura 4.2 mostra um exemplo de tradução de *links* de abstrato para concreto.

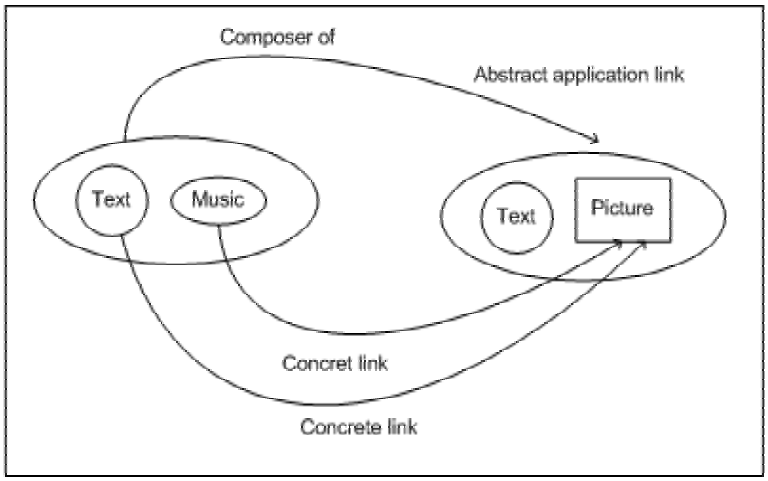


FIGURA 4.2 - Tradução de *links* de abstrato para concreto.
FONTE: Garzotto (1993).

Considerações: O HDM explora ainda várias outras situações mostrando como transformar situações reais em *links* que farão parte de aplicações de hipertexto e como entender e organizar estes *links* em documentos.

A limitação deste modelo é que é voltado somente a parte de levantamento e organização de requisitos. O HDM não traz especificações de interface, de estruturas, de tecnologias ou de resultados finais em *Web sites*.

4.2.2 - RMM

O *Relationship Management Methodology* (RMM) (Isakowitz, et all, 1995) faz uso de diferentes notações e primitivas de modelagem para expressar os mesmos conceitos, que podem ser observados na representação da estrutura de acesso "índice" no modelo RMM. Assim como o OOHDM e o HDM, o RMM também é influenciado pelo modelo Entidade-Relacionamento (E-R). O RMM reconhece três níveis, que são, o conteúdo, o hipertexto e a apresentação. O nível de conteúdo é modelado separadamente considerando que o nível de apresentação é refinado juntamente com o nível de hipertexto. Um formalismo dedicado modelando o que se chama de *Relationalship Management Data Model* (RMDM) é introduzido usando o modelo E-R para o nível de conteúdo e conceitos de propriedade os quais são influenciados pelo HDM para o nível de hipertexto e para o nível de apresentação.

O conceito chamado de *slices* (pedaços) é usado no plano entre o nível de conteúdo e de hipertexto dos atributos de entidades do diagrama E-R e/ou previamente definem-se os *slices* que são agrupados. Assim o modelo navegacional de índice e os derivados são criados.

As relações entre as entidades são usadas para capturar informações contextuais durante a navegação. O diagrama de aplicações cria a visão global no nível de apresentação das aplicações de Dados-*Web* capturando todas as páginas e *hiperlinks* pertinentes. Somente os aspectos estruturais são considerados em todos os níveis. O RMM especifica o processo de desenvolvimento com os passos iniciais para a análise de requisitos, modelando o conteúdo em forma de diagramas E-R. Estes são seguidos por passos iterativos referindo-se ao diagrama de aplicação, ambos, base e topo pelo *design* de *slice*.

A metodologia RMM destina-se ao projeto e ao desenvolvimento de aplicações hipermídia que possuem uma estrutura regular no domínio de interesse, como classes de objetos relacionados. Prevê também o uso de uma linguagem para especificar os objetos e os mecanismos de navegação em aplicações hipermídia. Para tanto, o modelo de dados RMDM faz uso de um conjunto de primitivas gráficas para descrever aplicações hipermídia como:

- **Primitivas de domínio E-R**: modela informações sobre o domínio da aplicação, tais como, entidades, atributos e relacionamentos.
- **Primitivas de domínio RMM**: atributos de uma mesma entidade podem ser agrupados em pedaços menores (*slices*) representando diferentes aspectos de um mesmo objeto. Cada *slice* pode ser visto como um nó ou uma página em um mesmo contexto.
- **Primitivas de acesso**: possibilitam a navegação através dos objetos da aplicação hipermídia. Ligações unidirecionais e bidirecionais podem ser utilizadas para acessar diferentes *slices* de uma mesma entidade, semelhante às ligações estruturais do modelo HDM. A navegação entre entidades diferentes (tipos de ligações em HDM) pode ser feita por meio de qualquer uma das primitivas abaixo ou por variações e/ou combinações entre elas:

  - **Índices**: o acesso é obtido a partir de uma tabela de conteúdos para uma lista de instâncias de entidades.

  - **Roteiro guiado**: consiste em um caminho linear através de uma coleção de instâncias de entidades.

- **Agrupamentos**: é um mecanismo semelhante a um menu de opções que oferece acesso a outras partes de um documento hipermídia.

O modelo de dados RMDM utiliza conceitos do modelo E-R como base para descrever a estrutura de uma aplicação hipermídia. A partir da estrutura inicial, as primitivas específicas são usadas para representar elementos específicos de hipermídia (*slices*, *links*, índices, roteiro guiado e agrupamentos) para modelar aplicações.

A abordagem RMM baseia-se no modelo cascata tradicional (fases de projeto, desenvolvimento e construção). Atividades tradicionais da engenharia de *software* devem ser empregadas antes e depois da modelagem da aplicação propriamente dita, que pode ser dividida em:

- **Projeto de entidades e relacionamentos**: são identificados as entidades e os relacionamentos mais significativos da aplicação.
- **Projeto de *slices***: determina como as informações da entidade escolhida serão apresentadas ao usuário e como podem ser acessadas através de ligações unidirecionais ou bidirecionais.
- **Projeto navegacional**: nesta etapa são definidos os caminhos que habilitarão a navegação hipertexto. Cada relacionamento obtido no modelo E-R é analisado e transformado (se necessário) em uma primitiva de navegação (Isakowitz, et all, 1995).

Considerações: O RMM mostra os recursos necessários para fazer o levantamento e a análise dos requisitos, bem como a transformação dos requisitos obtidos, em estruturas de hipertextos.

A limitação deste modelo, mesmo fazendo referências a *slices* e *Data-Web*, é estar muito direcionado ao desenvolvimento de *softwares* tradicionais. Isso faz com que aumente a distância na comunicação entre desenvolvedores e administradores de *Web sites*.

4.2.3 - OOHDM

O método *Object-Oriented Hypermidia Design Method* (OOHDM) (Rossi, et all, 1996), é um método para desenvolver hipertextos baseado na orientação a objetos; analisa os objetos a serem colocados nos hipertextos e suas ligações; é composto por quatro atividades diferentes.

- **Projeto conceitual**: modela a semântica do domínio da aplicação.
- **Projeto navegacional**: leva em consideração o perfil do usuário e a tarefa a ser executada; dá ênfase aos aspectos cognitivos. Apresenta ainda, uma reorganização do modelo conceitual utilizável por um grupo de usuários. O nível de hipertextos é modelado por dois conceitos diferentes, entendidos como "Classe de Navegação" e "Contexto de Navegação". As classes de navegação são feitas por "nós" e por "*links*". Os nós representam uma visão das classes conceituais, usando uma linguagem de consulta, enquanto os *links* são os relacionamentos exploráveis. O contexto de navegação é a estrutura do espaço de navegação.
- **Projeto abstrato da interface**: modela objetos perceptíveis, implementa as metáforas escolhidas e descreve a interface para os objetos navegacionais. Oferece a percepção que o usuário terá do espaço navegável, os diagramas de configuração e a visão abstrata dos dados.
- **Implementação**: esta atividade é responsável por transformar um projeto em algo real, fazendo uso das tecnologias escolhidas.

No OOHDM, não há formalismos específicos empregados, mas somente a referência que for estabelecida no diagrama. No nível de apresentação chamado de "Visualização de Dados Navegacionais", o contexto navegacional propicia o modelo navegacional como o início das opções de navegação. O comportamento da modelagem somente é mencionado a respeito da definição da seqüência da navegação por meio de restrições. No nível de apresentação, o "Modelo de Apresentação Estática" usa os diagramas da UML para representar as composições por meio de gráficos e descreve o *layout* de interface do usuário. O modelo de apresentação dinâmica emprega a UML estabelecida nos diagramas para descrever a ativação de navegação e as transformações de interface do usuário.

Para representar as interfaces de objetos mais freqüentemente usados, como texto, imagem, áudio e botões, são criados estereótipos (Rossi et all, 1996).

Considerações: O OOHDM trata de classes, contextos e transformações, fazendo uso da UML para estereotipar os modelos das fases consideradas necessárias por este modelo.

As limitações encontram-se na falta de instruções de implementação, tipos de linguagens e/ou tecnologias que possam ser utilizadas e o porte/tamanho do *Web site* que poderia fazer

uso deste tipo de modelagem. O OOHDM também não mostra se o resultado de um projeto melhoraria o sistema para o usuário, que vê/entende somente a arte e o *design* e não a engenharia usada no desenvolvimento do sistema.

4.2.4 - Conallen

O modelo Conallen (Conallen, 1999) é diferente dos demais mostrados nesta Seção, por ser em grande parte, uma tecnologia direcionada. A UML é empregada como um formalismo básico e estendido por meio de estereótipos e valores atribuídos. Ao invés de distinções entre os níveis de conteúdo, hipertexto e apresentação, o Conallen modela as páginas tanto do lado cliente como no lado do servidor, fazendo previsões de saída ou estereotipando as classes UML.

A UML em seu estado original, não possui estereótipos suficientes para modelar aspectos específicos de aplicações *Web*. Foi analisada, então, a WAE (*Web Application Extension for* UML), uma extensão para a UML criada por Jim Conallen (Conallen, 1999). A WAE estende a notação UML trazendo novos estereótipos com semântica e restrições adicionais que permitem a modelagem de elementos específicos da arquitetura envolvida numa aplicação *Web* incluindo-os nos modelos do sistema. Utilizando a WAE é possível representar páginas, *links* e o conteúdo dinâmico no lado cliente e no servidor.

A Figura 4.3 mostra um modelo cliente x servidor com estrutura de páginas dinâmicas.

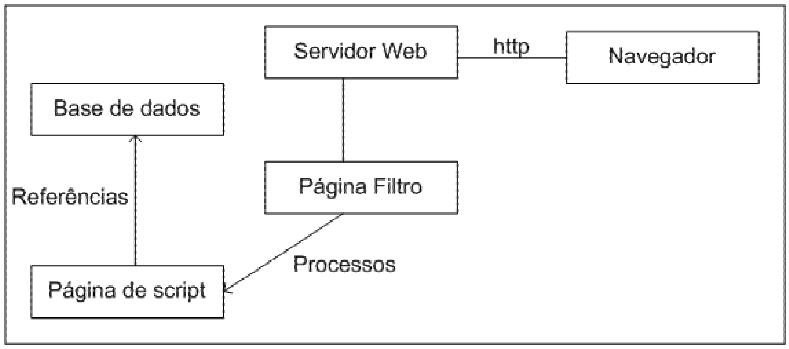


FIGURA 4.3 - Modelo cliente x servidor.
FONTE: Conallen (1999).

Associações estereotipadas são usadas para representar *hiperlinks* e para modelar o planejamento entre páginas clientes e páginas servidoras, desde que toda página dinâmica cliente, seja construída como uma página servidora. Dados que entram em formulários que

podem fazer parte de páginas clientes junto com seus relacionamentos de submissão para a página servidora são modelados por uma outra classe e associação de estereótipo, respectivamente.

O sistema Conallen mostra que a modelagem de um sistema pode ser representada por diferentes modelos consistentes entre si. Cada modelo tem uma audiência e um propósito específico. O modelo Conallen concentra-se em um modelo de *design* para aplicações *Web* com a audiência voltada para o *design* de uma arquitetura. Uma das considerações do Conallen é que se uma aplicação for modelada usando UML, uma página pode ser entendida como um objeto e isso gera questões como quais são as propriedades de tais objetos? Estas propriedades são apropriadas para representar os elementos de um *layout* como fontes, tabelas, textos etc? Os *scripts* de uma página poderiam ser identificados como métodos e uma página como objeto? Que modelo está sendo usado e quem faz a audiência?

Para este modelo de projeto o formato da interface do usuário é irrelevante e não necessariamente resulta em uma lógica do sistema. *Scripts*, especialmente *scripts* do lado do servidor resultam em um comportamento do sistema (e alguns sistemas representam um conjunto lógico do sistema). Além disso, não é difícil perceber variáveis em uma página de *script* como sendo atributos de uma página-objeto e função em uma página como sendo um método. Tudo isso é apropriado para um modelo de *design* e para uma aplicação *Web* voltada ao *design*.

Isso, no entanto indicia um outro problema: páginas *Web* podem conter *scripts* para a máquina servidora e para a cliente. Misturando atributos e métodos para execuções clientes e servidoras pode dificultar o entendimento e isso pode ser resolvido usando um benefício relativamente novo para a modelagem que são as novas extensões.

A Figura 4.4 mostra os ícones Estereótipos do Modelo Conallen.

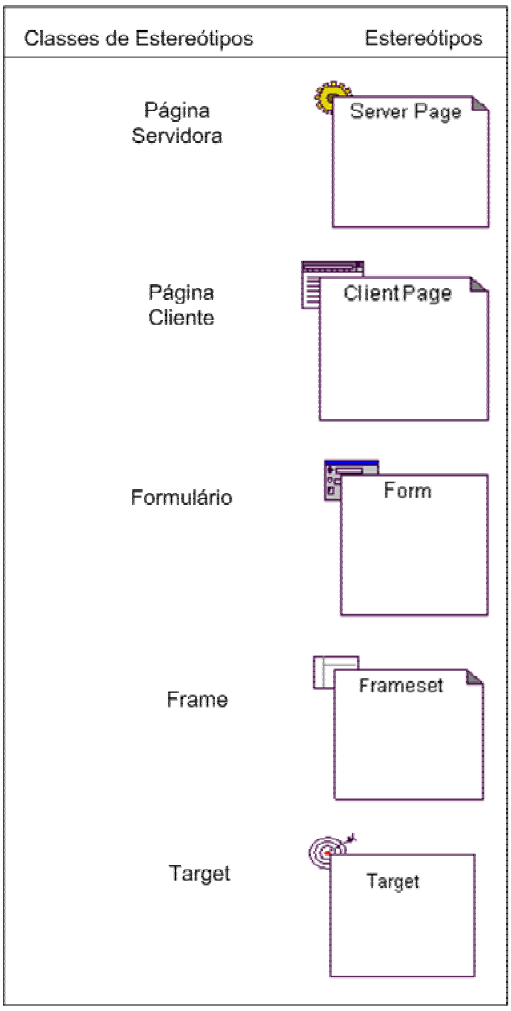


FIGURA 4.4 - Ícones estereótipos.

FONTE: Conallen (1999).

Parte das extensões do mecanismo da UML é capaz de especificar diferentes ícones para estereotipar classes. O problema das páginas *Web* que têm diferentes *scripts* e variáveis executadas no servidor e no cliente pode ser resolvido em uma destas duas formas. Na primeira seriam definidos os estereótipos do método cliente e do método servidor. Na página-objeto o método que executa no servidor será estereotipado como {{método servidor}} e as funções que rodam no cliente {{método cliente}}. Isso resolve o problema das distinções dos atributos e métodos das páginas-objeto, no entanto ainda continuam alguns pontos não resolvidos. As demais complicações surgem mais tarde quando as associações são feitas com outros componentes do modelo, onde não fica claro se alguns dos relacionamentos são válidos somente no contexto do método servidor ou no cliente.

A melhor maneira de modelar uma página é com duas classes de estereótipos separadamente. A Figura 4.5 mostra o estereótipo de uma página servidora construindo uma página cliente.

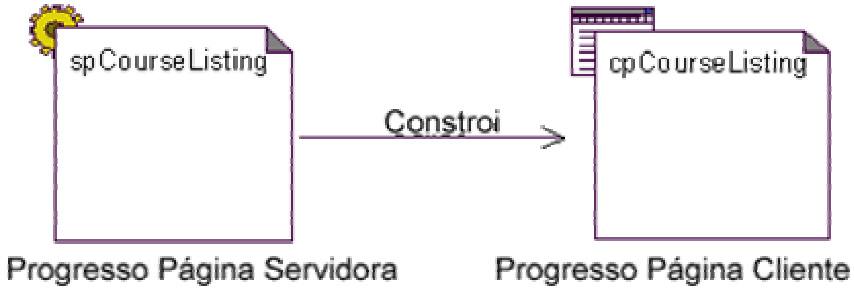


FIGURA 4.5 - Páginas cliente construindo uma página servidora.
FONTE: Conallen (1999).

Existem também estereótipos de classes para Java *Applets*, JavaScripts, controles *ActiveX* e *frames*. O Conallen não discute qualquer comportamento modelando a parte das operações as quais podem ser definidas junto às classes estereotipadas e não sugere nenhuma fase de modelagem (Conallen, 1999).

Considerações: O modelo Criado por Conallen (WEA-UML), entre os mostrados nesta Seção, é o que está mais voltado ao desenvolvimento de *Web sites*, pois trabalha resultados estáticos e dinâmicos de *Web sites*. Este modelo faz referência a páginas que são geradas dinamicamente como respostas de consultas a bancos de dados; faz uso dos recursos da ES e da UML e cria estereótipos específicos para representar situações que ocorrem tanto do lado servidor como do lado cliente em um processo de páginas dinâmicas. Dos modelos citados nesta Seção, o proposto por Conallen é o que mais se distingue dos modelos da Engenharia de *Software*, pois considera necessidades específicas para projetos *Web*.

A limitação deste modelo é que mesmo fazendo referências a resultados de páginas dinâmicas, clientes e servidoras, é um modelo que instrui o desenvolvedor a testar aplicações, interatividades e resultados obtidos de páginas clientes e servidoras. O modelo proposto por Conallen não é um modelo voltado ao usuário final que trata da usabilidade ou se preocupa em como os resultados dinâmicos serão vistos e entendidos pelo usuário. Uma outra limitação é que faz a modelagem somente de um segmento de *Web sites* que é a entrada e saída de dados em banco de dados e não do processo de desenvolvimento de *Web sites* como um todo.

4.2.5 - WebML

O *Web Modeling Language* (Ceri, 2000) é uma notação visual para especificar a composição e a navegação de aplicações de hipertexto. Construído com padrões populares como Entidade-Relacionamento e UML, o *Web Modeling Language* permite especificar complexas aplicações *Web* em plataformas diferentes.

O WebML estabelece graficamente especificações formais incorporadas em um processo completo de *design* o qual pode ser visto através de ferramentas visuais. Os principais objetivos do processo de *design* do WebML são:

- Ilustrar a estrutura de uma aplicação *Web* com um alto nível de descrição, o qual pode ser usado em consultas, atualizações e manutenções;
- Propiciar múltiplas visualizações do mesmo contexto;
- Separar informações que contêm a composição das páginas, navegação e apresentação, as quais podem ser definidas e organizadas de forma independente.
- Registra meta-informações coletadas durante um processo de *design* dentro de um repositório, o qual pode ser usado enquanto o tempo de vida da aplicação for dinamicamente gerando páginas *Web*;
- Modelar usuários e comunidades explicitamente em um repositório, para permitir políticas de permissões e especificações para aplicações *one-to-one*;
- Habilitar especificações de manipulação de dados e operações para atualizações de conteúdos ou interações com serviços externos arbitrários.

Os modelos de composição, estrutura, navegação e apresentação possibilitam a descrição de leitura em *Web sites*. Estes modelos podem ter ampliações com especificações de gerenciamento do conteúdo e integração com serviços externos, durante operações adicionais, as quais podem ser definidas e adicionadas ao modelo de hipertexto. Os modelos são vistos como um efeito de navegação e permitem uma especificação que normalmente encontra interação como padrão de entrada de dados, gerenciamento de dados pessoais e carrinhos de compras.

Os modelos previstos no WebML são classificados como modelo de dados, modelo de hipertextos, modelo de apresentação e processos WebML.

- **Modelo de dados**: o modelo de dados WebML é uma adaptação do modelo conceitual para o *design* de dados, como já é mostrado na Engenharia *Web*, tal

como *design* de banco de dados, engenharia de *software* e demais representações de conteúdo. O modelo de dados é compatível com o modelo Entidade-Relacionamento, usado no *design* conceitual de banco de dados, com os diagramas de classes da UML, usado na modelagem de orientação a objetos. O fundamento dos elementos do modelo de dados são **entidades**, definidas como recipientes de elementos de dados, e **relacionamentos**, definidos como semânticas de conexões entre as entidades. As entidades têm propriedades nomeadas, chamadas atributos, com tipos associados. As entidades podem ser organizadas em gêneros hierárquicos e os relacionamentos podem ser limitados pelo significado da cardinalidade das restrições. As instâncias das entidades são consideradas individualmente e endereçadas significando um "Identificador Único". Os Identificadores Únicos da WebML são conceitos abstratos, os quais podem ser implementados de forma alternativa subjacente ao recipiente de gerenciamento com chave primária para armazenamento de dados relacionais ou atributos ID de XML em fontes de dados XML.

A Figura 4.6 mostra um exemplo do modelo de dados representando informação sobre álbuns musicais que são compostos por artistas, sobre os quais são disponibilizadas algumas resenhas. Cada álbum pode conter várias faixas.

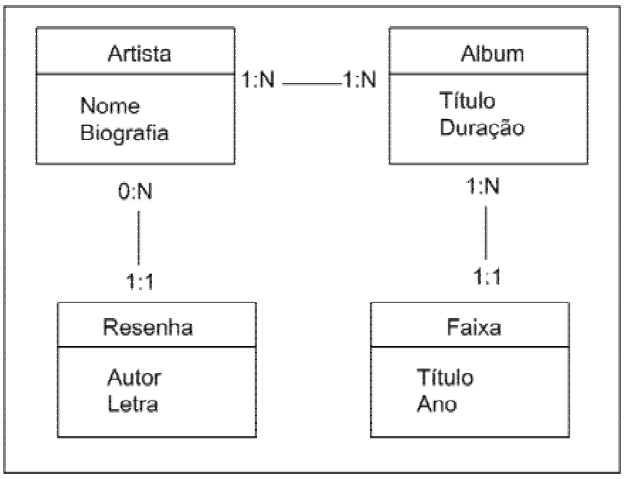


FIGURA 4.6 - Modelo de dados WebML.
FONTE: Ceri (2000).

- **Modelo de hipertextos**: o modelo de hipertextos especifica a composição e a navegação do *site*. A composição descreve quais páginas compõem o hipertexto e quais unidades de conteúdo formam uma página. As páginas do *Web site* são

recipientes da informação mencionada no cabeçalho. As unidades são atômicas contendo elementos usados para publicar informações descritas no modelo de dados. Sete tipos de unidades são pré-definidas em WebML para compor as páginas: **dados**, **multi-dados**, ***index*** (e suas variáveis de múltipla escolha e hierarquias), **entradas** e ***scrollers***. Cada unidade está associada a uma entidade subjacente da qual a unidade de conteúdo é computada. A especificação da entidade subjacente impõe o tipo de objeto do qual a unidade de conteúdo é derivada. Quando apropriado, as unidades podem ser opcionalmente associadas a um seletor e uma especificação do grupo de restrições determina a instância atual da entidade subjacente para ser usada como uma unidade de conteúdo em tempo corrente.

A **navegação** do *site* é especificada diretamente nos *links*. Os ***links*** podem ser definidos entre as unidades dentro de uma página comum, entre unidades localizadas em páginas diferentes e entre páginas. A informação trazida junto do *link* é chamada de navegação contextual ou simplesmente contexto. Os *links* que trazem informação de contexto são chamados de *links* contextuais, considerando que os *links* que não têm contextos informativos associados são chamados de *links* não-contextuais. Uma informação de contexto é tipicamente necessária para assegurar a computabilidade das unidades.

A Figura 4.7 refere-se a um exemplo de especificação de hipertexto do WebML.

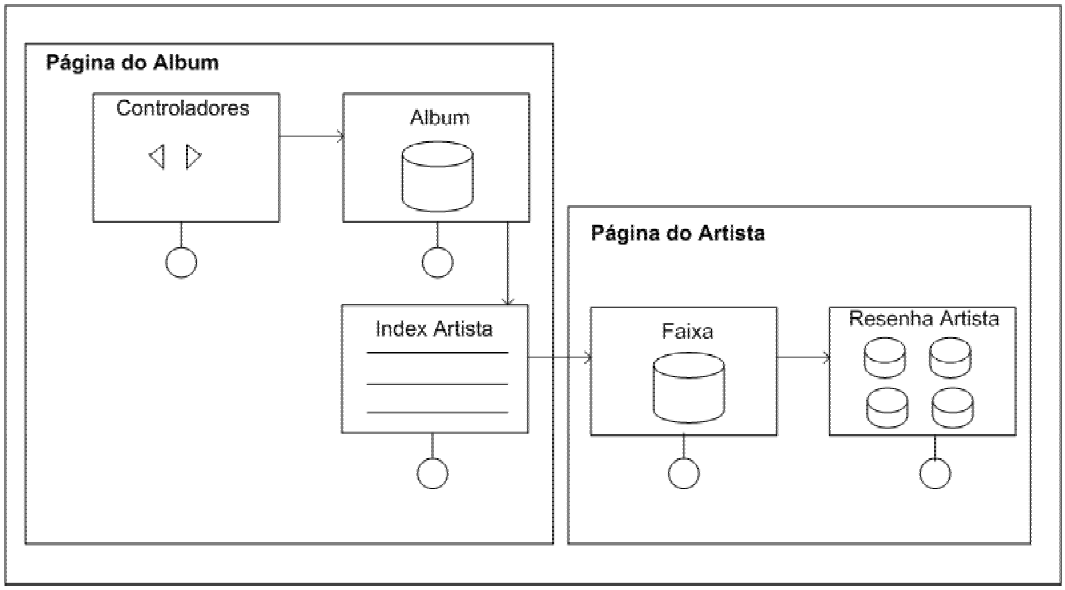


FIGURA 4.7 - Especificação de hipertexto do WebML.

FONTE: Ceri (2000).

- **Modelo de apresentação**: a apresentação é uma tarefa ortogonal definindo a visualização das páginas dentro da visão do *site*. Desde que as especificações WebML possam ser representadas usando XML, a apresentação é considerada como um documento de transformação, mapeando as especificações WebML da página escrita em uma concreta implementação de linguagem como JSP ou ASP.NET. Conseqüentemente a apresentação é endereçada em WebML, utilizando estilos XSL para visualização do *site*, páginas, unidades e unidades de sub-elementos. As folhas de estilo XSL inserem especificações WebML, codificando como documentos XML conforme a definição do documento WebML e exporta modelos de páginas incorporando o código e os dados acessados naquela consulta. A implementação de WebML pode incluir vários estilos de apresentação pré-definidos e os componentes do lado do servidor, suportando o acesso a consultas de dados necessários para compor o conteúdo das páginas produzidos pelas folhas de estilo XSL.

- **Processos WebML**: o WebML aborda a desenvolvimento de aplicações *Web* consistindo em diferentes fases que podem ser aplicadas de maneira interativa e incremental. O processo suporta vários ciclos, cada ciclo produzindo um protótipo ou uma versão parcial da aplicação que permite a condução de testes e a evolução desde o desenvolvimento das fases iniciais.

  A Figura 4.8 mostra o desenvolvimento de aplicações *Web* consistindo em diferentes fases que podem ser aplicadas de maneira interativa e incremental.

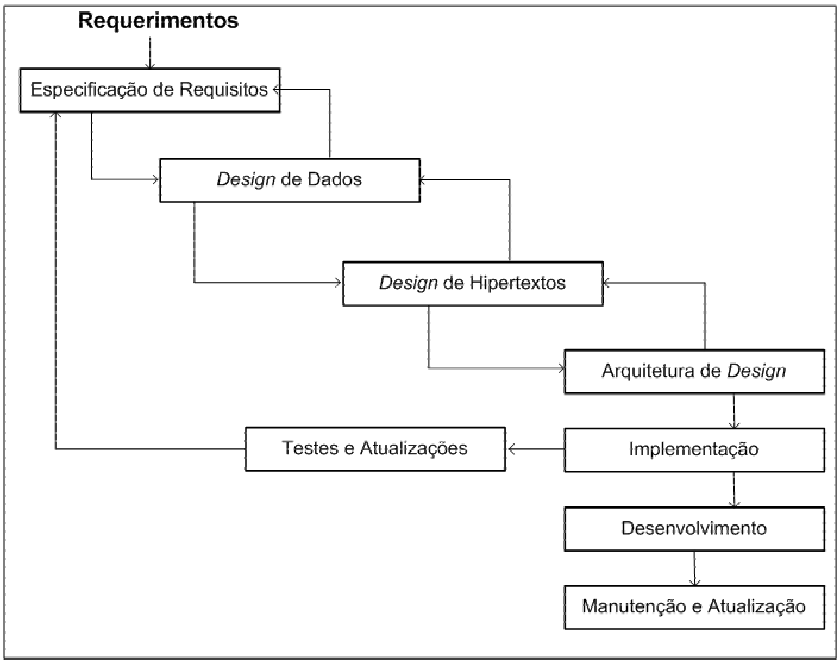


FIGURA 4.8 - Desenvolvimento de aplicações *Web*.

FONTE: Ceri (2000).

Fora do processo como um todo, os dados e o *design* de hipertexto são as atividades mais influenciadas pela adoção do WebML. A aproximação do *design* de dados não aponta uma substituição para as diretrizes de proposta geral do modelo de dados conceitual, mas simplesmente os estende com algumas poucas regras específicas de manuseio, os quais podem ajudar no *design* de dados para a Web. A publicação de dados e o gerenciamento do conteúdo de aplicações têm algumas regularidades peculiares, as quais podem ser exploradas no *design* de dados. O reconhecimento destes detalhes pode ajudar no *design* dos dados organizando o trabalho de uma maneira sistemática, a qual normalmente resulta em sistemas de dados mais consistentes. Entretanto, este método enfatiza regras distintas representadas por objetos e usa estas distinções para propor uma seqüência de passos para agrupar esquemas de dados de aplicações *Web*. Depois do *design* de dados, o *design* de hipertexto procede fazendo com que o primeiro *design* das atividades aponte para a produção do primeiro nível de especificação da visualização do *site* que explora uma notação textual informal. Para expressar a visibilidade dos níveis de visualização das áreas do *site* (marcas, padrões ou áreas internas) especifica o conteúdo da área usando as entidades e relacionamentos do esquema de dados. Procedendo dessa maneira, uma atenção especial destina-se as regras representadas

por vários elementos de dados, os quais podem ser explorados acessando informações, para publicar conteúdos retirados dos objetos, para interconexão retirada dos objetos ou para personalização de propósitos. Um *design* detalhado explora sub-esquemas de hipertexto. O *design* detalhado explora os sub-esquemas de hipertextos do WebML, os a quais são configurações "canônicas" de páginas, unidades, construção, acesso, interconexão e personalização dos sub-esquemas.

Considerações: O WebML apresenta basicamente um modelo de dados, um modelo de hipertexto, um modelo de apresentação e um modelo de processos.

O modelo de dados é representado tipicamente pelo modelo Entidade-Relacionamento, usado na Engenharia de *Software*. O modelo de hipertexto ilustra o sistema de hipertexto e mostra as possibilidades de respostas dos bancos de dados e como o usuário pode interagir com as páginas. O modelo de apresentação mostra o *design* das páginas dentro de um *site* com o exemplo do uso de linguagens do lado cliente e do lado servidor como recurso de implementação. O modelo de processos aborda diferentes fases que podem ser aplicadas de maneira incremental. O processo mostra vários ciclos através de protótipos de versões parciais de uma aplicação.

As limitações do WebML encontram-se no uso de seus conceitos. Por ser um modelo conceitual de modelagem, mesmo mostrando como fazer uso de folhas de estilo e XML, não prevê uma forma de implementação. No modelo de processos, por exemplo, o *design* dos dados, o *design* de hipertexto, aparecem logo no segundo e terceiro nível e a implementação, que dentro de uma seqüência lógica de aplicações para a *Web* apareceria logo após a especificação dos requisitos, aparece em um quarto nível. Dentro da análise realizada no WebML não foi possível identificar uma forma de implementação utilizando a seqüência sugerida pela modelagem se o *design* dos dados, o *design* de hipertexto e a arquitetura do *design* que aparecem antes de se começar a implementação; que tipos de testes de evolução podem ser feitos antes do efetivo desenvolvimento; como se pode definir *design* de dados, *design* de hipertextos e arquitetura de *design* sem antes fazer o planejamento da estrutura do *Web sites*.

4.2.6 - Tabela Demonstrativa dos Modelos Citados

TABELA 4.1 - Comparativo entre os modelos apresentados nesta Seção.

Modelo	Ano	Autor	Fonte	Resumo
HDM	1993	Garzotto, França	ACM *transaction on Information Systems*. Vol. 11.	Dispositivo de modelagem para descrever aplicações de hipertexto. Define semânticas de navegação que especificam a dinâmica do comportamento de um sistema hipermídia.
RMM	1995	Isakowitz, Tomás	*Communications* of ACM. Vol. 38.	Faz uso de diferentes notações e primitivas de modelagem para expressar os mesmos conceitos; é influenciado pelo modelo E-R; reconhece três níveis: hipertexto, apresentação e conteúdo.
OOHDM	1996	Rossi, Gustavo	PUC Rio: tese de doutorado.	Método para desenvolver hipertextos, baseado na orientação a objetos; analisa os objetos a serem colocados nos hipertextos e suas ligações; usa as possibilidades da UML para representar as composições navegacionais.
Conallen	1999	Conallen, Jim	*Communications* of ACM. Vol. 42.	Modela páginas no lado cliente e no lado servidor, estereotipando as classes UML; Sugere estereótipos específicos para aplicações *Web*; associações estereotipadas são usadas para representar *hiperlinks*.
WebML	2000	Ceri, Stefano	WWW9 *Conference, Amsterdam*, 2000.	Notação para visualizar especificações complexas de *Web sites* no nível conceitual. Todos os conceitos do WebML são especificados graficamente e em XML; a composição das abstrações são baseadas em um restrito número de componentes de hipertexto os quais são montados em uma página e interconectados por *links*.

4.2.7 - Modelos Desenvolvidos a Partir dos Modelos Mostrados nesta Seção

Além dos modelos mostrados nesta Seção, existem outros que são voltados à modelagem de dados em projetos para a *Web*, que vieram a partir dos modelos citados, como por exemplo, o OOHDM-ML e o Modeling data entry and operations in WebML.

OOHDM-ML: (Medeiros, 2001) apresenta uma linguagem declarativa para especificação de aplicações hipermídia projetadas de acordo com as primitivas do método OOHDM, e o ambiente OOHDM-XWeb, criado para prover a implementação dessas aplicações. Compreende duas etapas principais: a especificação da aplicação através de um método de projeto (por exemplo, OOHDM) e sua implementação em alguma linguagem (através de um ambiente de suporte). A Linguagem XML é utilizada é utilizada para descrever o projeto OOHDM para uma aplicação hipermídia.

Modeling data entry and operations in WebML: (Ceri, 2002) faz uso dos recursos WebML com entradas de dados e unidades de operações para reunir informações dos clientes e chamar operações arbitrárias. Operações pré-definidas também são sugeridas como construtor interno de primitivas para suporte de padrões de atualizações do conteúdo subjacente da fonte de dados (representada como E-R). As extensões primitivas do WebML permitem visualizar a modelagem de páginas *Web* integrando acessos de escrita e leitura em aspectos essenciais como aplicações de *e-commerce* (incluindo perfil de usuários e gerenciamento de carrinhos de compra).

4.2.8 - Considerações Gerais

Os *Web sites* encontram-se em fase de maturidade ou na fase onde muitos problemas existentes estão sendo identificados, mas com poucas soluções apontadas e as soluções já apontadas, muitas vezes, são apenas teóricas. As dificuldades de um *Web designer* para transformar um projeto feito a partir da maioria dos modelos de Engenharia *Web* em um produto final ou em *Web site*, ainda são muito significativas, devido a falta de seqüências lógicas, que possam ser colocadas em prática.

O processo ora proposto considera que *Web sites* são sistemas desenvolvidos por profissionais da *Web*, para usuários da *Web*. Dessa forma não bastaria um modelo que apenas fizesse a modelagem de necessária à implementação se os benefícios desta modelagem não chegassem até o usuário. Também não bastaria um *design* atraente e um bom sistema de

hipermídia, com estrutura e funcionalidade inconsistentes. Este processo propõe três fases para o desenvolvimento de *Web sites* que se compreendem pelo levantamento de requisitos, implementação e *design*, fazendo uso de técnicas de *design*, linguagens, tecnologias e dos recursos da UML.

CAPÍTULO 5

PROCESSO DE DESENVOLVIMENTO DE *WEB SITES* COM RECURSOS DA UML

Nesta Seção, introduz-se um processo de desenvolvimento de *Web sites*, que é a contribuição deste trabalho. Este processo de desenvolvimento, chamado de "Processo de Desenvolvimento *Web* com recursos da UML" (PDW-UML), especifica três fases, mostrando a evolução de um *Web site* desde o levantamento de requisitos até o *design* final das interfaces. Como o processo abrange diversas etapas e as mostra em três fases, pretende-se com isso reduzir as dificuldades existentes pelos profissionais da *Web* em transformar os projetos *Web* em um produto final, que são os *Web sites*.

Este processo mostra a solução para alguns problemas de interfaces *Web* mostrados pela Engenharia *Web* como estrutura, tempo de carregamento de cada página, tamanho de URLs e *design* das interfaces procurando colocar em prática o que é chamado pela Engenharia *Web* de "Sistema Orientado ao Usuário" (Leite, 2002).

Para o desenvolvimento de um *Web site* com o PDW-UML, pode-se utilizar um Sistema Operacional como o Windows NT/2000/XP que utiliza o servidor IIS, sendo necessária a instalação de um servidor de PHP (poderia ser um outro servidor de páginas dinâmicas. Para as explanações deste trabalho será usado PHP). Outra opção são os sistemas Unix/Linux com o servidor Apache.

Diferentes dos *softwares* convencionais onde se estudam os recursos de *software* e *hardware* disponíveis em uma organização antes de se começar a desenvolver um produto, os *Web sites* têm que ser desenvolvidos considerando "possibilidades de recursos". Por isso, precisam buscar padrões de desenvolvimento que possam reduzir ao máximo a dependência dos recursos do usuário, fazendo com que uma página seja vista da maneira mais uniforme possível, independente do sistema do usuário.

Uma solução encontrada para esta necessidade é mostrar no presente processo um tipo de estrutura que utiliza a tecnologia *Server Side Include* (SSI - Seção 2.2.2.1) onde a máquina servidora seja responsável pela página gerada e não os recursos da máquina do usuário. Quando um usuário acessa uma página desenvolvida com SSI, a máquina cliente faz a requisição para a máquina servidora que processa a requisição, enviando para o usuário

somente o HTML gerado. Assim, a resposta será a mesma para qualquer usuário, independente dos recursos que possa ter em sua máquina.

No PDW-UML não há o reconhecimento da necessidade de desenvolvimento de protótipos e nem se determina uma fase específica para testes. Como a maior parte do funcionamento se concentra no sistema de navegação utiliza-se o procedimento de Implementar-Testar (IT) com ênfase nos testes no final das fases de implementação e de *design* das interfaces. Com isso identificam-se os riscos e as inconsistências logo no início do ciclo de vida link-página, ou seja, tão logo seja desenvolvida uma página e um *link* que aponte para esta, são feitos os testes necessários.

Este processo é baseado em três fases: a fase de levantamento de requisitos, a fase de implementação e a fase de *design* das interfaces.

- **Fase 1: levantamento de requisitos**. A fase de levantamento de requisitos levanta os objetivos do domínio e busca as informações disponíveis para que haja um planejamento do *Web site*; faz o levantamento das necessidades do projeto e do sistema através dos recursos da UML e organiza os requisitos necessários à implementação. Para fazer a modelagem o PDW-UML utiliza os diagramas da UML.

- **Fase 2: implementação**. a fase de implementação desenvolve os arquivos e diretórios que formam a arquitetura lógica de acordo com um modelo de Estrutura Dinâmica (ED) proposto para implementar o PDW-UML. Os requisitos levantados na fase anterior são analisados e utilizados para a composição das interfaces e do sistema navegacional. Nesta fase faz-se a previsão dos níveis de *Uniform Resource Locator* (URLs) e das portas de entrada do usuário; faz-se as convenções de *design* das interfaces; elabora-se a documentação com as informações pertinentes ao *Web site* e as instruções de administração para os *Web designers* e fazem-se os testes navegacionais a partir da máquina servidora.

- **Fase 3: *Design* das interfaces**. A fase de *design* das interfaces busca aprimorar o que foi desenvolvido nas fases anteriores propondo um tempo e um espaço dentro de um processo para que haja a realização artística e criativa. A proposta para a fase de *design* é uma forma de cultura que envolve estética, arte e ética como base para um estilo de *design*. É na terceira fase que se aprimoram os resultados do trabalho

realizado nas fases 1 e 2. Nesta fase o que predomina é a arte e a estética que podem ser vistas na distribuição dos *links* nas páginas, nos títulos, nos cabeçalhos, nos nomes dos *links*, nas cores, nas animações e no *design* em geral. A fase de *design* das interfaces é considerada como artística considerando que após a conclusão das fases 1 e 2 já se obteve a funcionalidade desejada. A partir daí o resultado pode ser sempre diferente de acordo com a maturidade (técnica/visual/auditiva/cultural) de cada profissional.

A Figura 5.1 mostra um diagrama com uma visão geral do processo proposto.

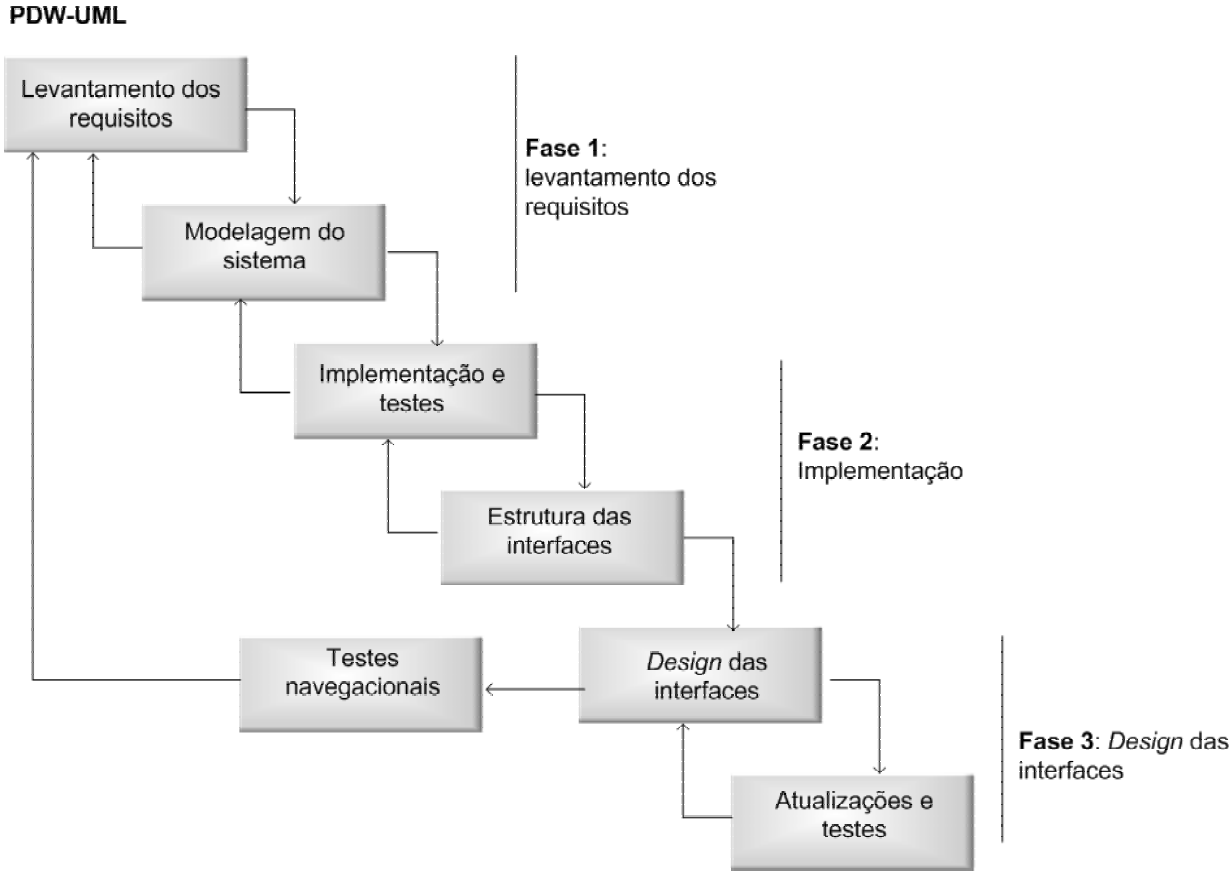


FIGURA 5.1 - Visão geral do PDW-UML.

O PDW-UML é um processo voltado ao desenvolvimento de interfaces, cujo funcionamento é baseado em uma forma de estrutura dinâmica que faz uso de HTML, SSI e CSS; o contexto geral do processo é baseado em três fases e a modelagem é baseada nos recursos da UML.

A seguir são mostradas as três fases do PDW-UML, que são as fases de levantamento de requisitos, implementação e *design* das interfaces.

5.1 - Fase 1: Levantamento de Requisitos

Na primeira fase o PDW-UML faz a análise dos requisitos necessários ao desenvolvimento do *Web site* e o levantamento dos requisitos de conteúdo. Nesta fase os requisitos de conteúdo são armazenados e classificados conforme perspectivas de desenvolvimento de interfaces. Na segunda fase, que é a fase de implementação, os requisitos de conteúdo são selecionados e utilizados conforme estejam de acordo com as informações que se pretende disponibilizar em cada interface.

No PDW-UML os requisitos são classificados como requisitos de conteúdo, requisitos operacionais e requisitos de desenvolvimento.

5.1.1 - Requisitos de Conteúdo

Os requisitos de conteúdo são os assuntos que serão expostos nas interfaces representando os atributos das mesmas (textos, imagens, *links* e formulários). Para obter os requisitos necessários para compor as interfaces é necessário o levantamento de assuntos que possam compor o que está sendo sugerido por um *link* como, por exemplo, se o domínio de uma empresa tem um *link* chamado:

- **Produtos**: para "produtos" é necessário fazer o levantamento dos produtos desta empresa com informações como preço, disponibilidades de estoque e de entrega, informações técnicas (tamanho, medidas, fragilidades), procedimentos de aquisição, descrição e informações complementares.
- **Serviços**: para "serviços" são necessárias informações sobre os serviços oferecidos pela empresa como, por exemplo, se são serviços de consultoria, suporte técnico, quais são as bases de preços para estes serviços, informações dissertativas sobre os serviços oferecidos, informações sobre a qualificação das pessoas que prestam estes serviços etc.
- **Funcionários**: para "funcionários" faz-se o levantamento de informações como qualificação profissional, canais de contato (telefone, *e-mail*, *instant messanger*, endereço para correspondência) etc.

- **Informações**: as interfaces de informações normalmente oferecem opções para que um usuário solicite informações que não estejam disponíveis no *site* ou então, para que o usuário forneça suas informações através de um sistema de cadastro. Para este tipo de interface os requisitos necessários são as informações de como o usuário fará uso dos formulários através de etiquetas que vêm antes dos campos de texto como, por exemplo, nome, sobrenome, *e-mail* etc, ou instruções de como solicitar informações ao administrador do domínio.

Estes requisitos podem ser provenientes de várias fontes como, por exemplo, textos de jornais, livros, revistas, áudios, vídeos, depoimentos, entrevistas, destaques publicitários, imagens, páginas *Web*, sugestões do cliente etc. A partir destas fontes são elaborados textos, imagens e *links* que compõem as interfaces.

Nesta fase todos os tipos de informações referentes ao domínio são aceitos e armazenados e posteriormente analisados para compor o conteúdo das interfaces. A seleção é feita durante a composição dos arquivos, na fase de implementação. Os requisitos de conteúdo são depositados nos diretórios correspondentes às interfaces. Os tipos de requisitos podem ser textos, áudios, imagens, vídeos ou páginas *Web*.

A Figura 5.2 mostra um exemplo de armazenamento das informações pertinentes a um domínio genérico aqui chamado de ”empresa”.

FIGURA 5.2 - Armazenamento de informações.

Antes de começar a implementação, os requisitos obtidos são armazenados em um diretório qualquer. Conforme haja segmentos pertinentes ao domínio, criam-se subdiretórios para que as informações possam ser armazenadas conforme um segmento que tenha perspectiva de formar um *link* e sua respectiva interface. Nesta fase entende-se que para cada subdiretório será desenvolvido um *link* que irá compor a barra de navegação. Conforme mostrado na

Figura 5.2, considera-se um domínio chamado “empresa”. Esta empresa oferece “produtos”, “serviços”, tem “funcionários” e dispõe de um canal de comunicação chamado de “informações”. O armazenamento dos requisitos em quatro subdiretórios representa uma perspectiva para o desenvolvimento de cinco interfaces: a *home page*, produtos, serviços, funcionários e informações. Estes requisitos continuarão como perspectivas de *links* e de interfaces até a segunda fase onde se inicia a implementação.

5.1.2 - Requisitos Operacionais

Para que os requisitos de conteúdo possam ser organizados e para que se possa fazer a escolha dos requisitos operacionais é necessário que sejam definidos alguns aspectos funcionais como a arquitetura física e lógica, a forma de organização das informações, a estrutura onde será feito o *design* e a previsão de saída de banco de dados (geração de páginas dinâmicas) e os recursos de *hardware*, *software* e banda.

- **Arquitetura**: a arquitetura de um sistema é entendida como arquitetura física ou lógica, sendo que é necessário a existência da arquitetura física para que a lógica possa existir, ou seja, a arquitetura lógica é construída a partir da arquitetura física.

  - **Arquitetura física**: a arquitetura física representa os recursos de *hardware* existentes tanto na máquina servidora como na máquina cliente.

  - **Arquitetura lógica**: a arquitetura lógica representa a maneira como os arquivos e diretórios estão organizados a partir de um diretório *root*.

  A Figura 5.3 mostra a ilustração de uma arquitetura física e de uma arquitetura lógica.

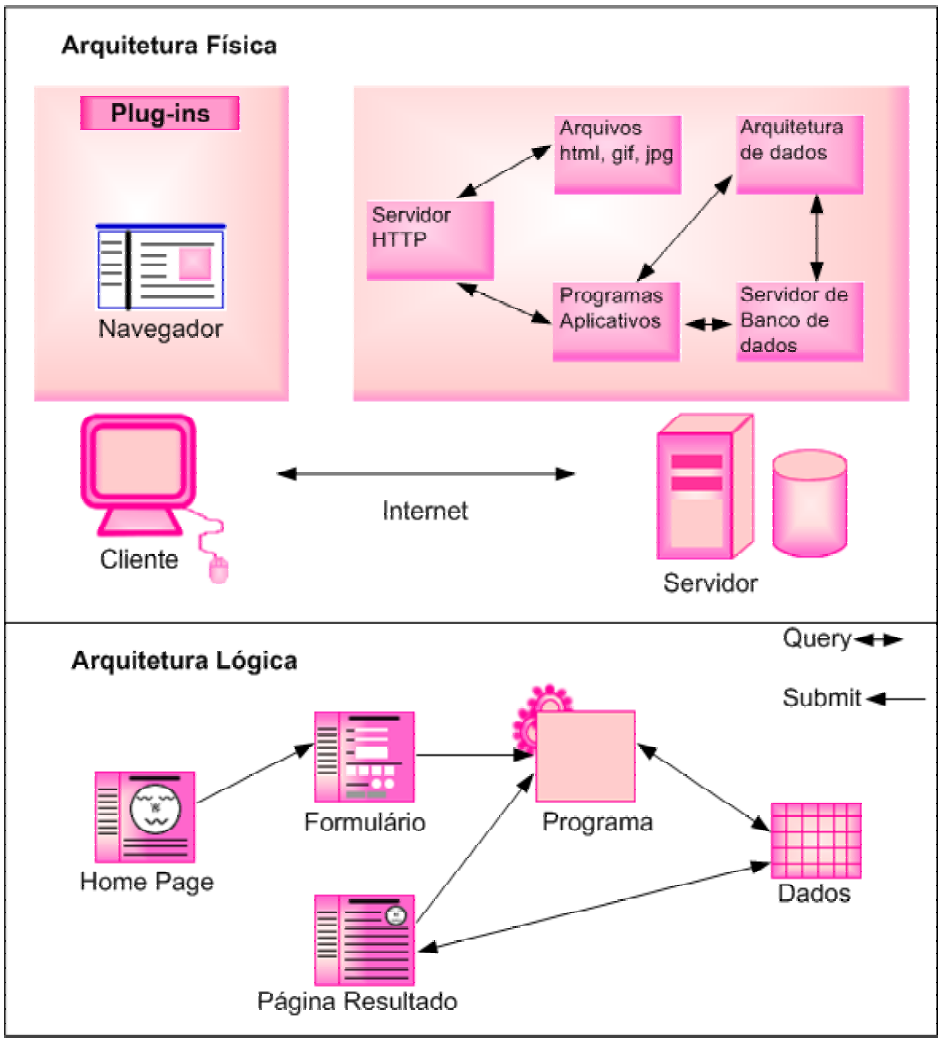


FIGURA 5.3 - Arquitetura física e arquitetura lógica.

- **Organização das informações**: de acordo com a arquitetura lógica têm-se as classificações das formas de organização das informações. Essas formas de organização podem ser hierárquica, linear e em forma de rede. As organizações linear e em forma de rede quase não são usadas em *Web sites*, pois não conseguem proporcionar um bom entendimento, bem como uma boa organização do sistema. A organização hierárquica é a que mais se utiliza atualmente. O número de arquivos existentes no diretório *root* dependerá da estrutura escolhida para as interfaces, independente de ser a organização hierárquica ou outra forma de organização. Para o sistema de estrutura proposto no PDW-UML são necessários dois arquivos no diretório *root* e os demais em seus respectivos subdiretórios.

  A Figura 5.4 mostra uma organização hierárquica, desenvolvida a partir de um diretório *root*.

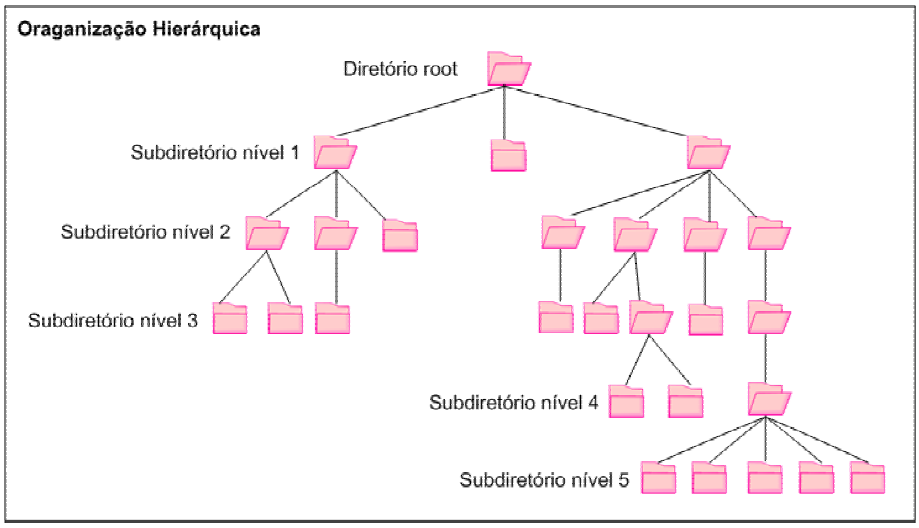


FIGURA 5.4 - Organização hierárquica.

- **Estrutura onde será feito o *design***: a estrutura escolhida é que determina a forma de navegação do *Web site*. De acordo com a estrutura escolhida é que se pode decidir quantos arquivos ficarão no diretório *root*; qual o valor do parâmetro "*target*" que será utilizado nos *links*; quantos níveis existirão nos URLs e a criação de diretórios específicos para um determinado tipo de arquivo. A distribuição do conteúdo dentro de um arquivo também é feita de acordo com estrutura escolhida.

  Para interfaces com Estrutura Dinâmica (ED), como propostas neste trabalho, elaboram-se os arquivos de forma independente e faz-se a inclusão dos que sejam necessários para compor uma página em um arquivo-matriz que inclui os arquivos de interface e de conteúdo dentro do próprio arquivo que faz a estrutura.

- **Previsão da interface de saída de banco de dados**: a estrutura também determina como será a interface gerada dinamicamente através da saída obtida de consultas a bancos de dados. Para o sistema de ED basta indicar o nome do arquivo-matriz elaborado para a saída da interface que será gerada para cada tipo de consulta para que se obtenha o conteúdo gerado juntamente com o menu e as instruções de interface.

- ***Hardware***: os recursos de *hardware* representam os recursos físicos que uma máquina precisa ter para que os *softwares* sejam instalados. Para o desenvolvimento e administração de um *Web site*, os recursos que mais influenciam no rendimento operacional são espaço em disco, memória, processador e placa de vídeo. Os demais

recursos dependem de necessidades específicas ou necessidades que um *Web site* poderá ter e outro não, como por exemplo, placa de som, *Web* câmera etc.

- ***Software***: os *softwares* necessários ao desenvolvimento e administração de um *Web site* podem ser definidos como o sistema operacional, o servidor de páginas *Web*, *softwares* de criação e tratamento de imagens, edição de códigos, diagramação, edição de textos e de animações. Algumas linguagens dependem de *softwares* específicos para que sejam processadas ou compiladas, como é o caso da linguagem PHP que precisa do servidor Apache para ser processada.
- **Banda**: no lado cliente não há necessidades de grandes preocupações com a velocidade da banda, uma vez que os arquivos e diretórios são desenvolvidos na máquina local ou cliente e podem ser enviados ao servidor até mesmo de uma outra máquina onde haja um sistema de banda larga. No entanto, a velocidade do lado servidor é um dos fatores prioritários para determinar a velocidade de navegação (*upload* dos arquivos).

5.1.3 - Requisitos de Desenvolvimento

Os requisitos de desenvolvimento representam a definição das pessoas necessárias ao desenvolvimento do *Web site*; os cronogramas de entrega de cada fase, bem como da conclusão do *Web site*; os custos que incorrerão durante todo o processo de desenvolvimento e as previsões de custo de administração.

- **Pessoas**: o perfil dos profissionais necessários ao desenvolvimento e manutenção de um *Web site* varia de acordo com sua a complexidade. As categorias de profissionais da *Web* são relativamente novas, sendo que para profissionais que fazem o desenvolvimento e administração de *Web sites* os mais comuns são, *Web designers* e *Web masters*.

  - *Web designer*: é o profissional que cuida da parte visual do *site* como, por exemplo, desenvolver e manipular artes gráficas, bem como programar em linguagens como HTML, CSS e JavaScript, PHP etc.

  - *Web master*: é o responsável pelo desenvolvimento, manutenção e gerenciamento do *Web site*. Deve ter os mesmos conhecimentos que tem o *Web designer*, programação em linguagens diversas para a *Web*, técnicas de

redação e ainda conhecimentos da área técnica como TCP/IP, *hardware*, segurança de redes e preparação e segurança de *hosts*.

Dependendo da complexidade ou das necessidades do *Web site*, são necessários profissionais que ofereçam trabalhos especialistas como em bancos de dados e programação especializada em uma determinada linguagem/tecnologia.

- **Cronogramas**: os cronogramas podem ser desenvolvidos visando a conclusão do *Web site* como um todo ou visando a conclusão de fases específicas.
- **Custos**: os custos com um *Web site* são custos com pessoas, *softwares* e *hardwares*, nas fases de desenvolvimento e manutenção. Os custos variam de acordo com a qualidade de um trabalho, qualificação dos profissionais envolvidos, e com a complexidade do *Web site* em questão. É difícil generalizar a distribuição dos custos, pois estes variam de acordo com a complexidade e com o objetivo de cada projeto. Um *Web site* informativo que tenha como conteúdo predominante a recuperação de manuscritos, levantamento de fatos históricos, concentraria maiores custos na fase de levantamento de requisitos. Um *Web site* onde o objetivo principal seja o comércio eletrônico, tem os custos concentrados na fase de implementação devido as necessidades de segurança, cadastros de dados e testes de funcionalidades. Já um *Web site* com motivos artísticos como lojas de decorações, cosméticos e produtos em geral que precisam ser vistos a partir de um *design* diferenciado, concentra os maiores custos na fase final que é a fase de *design* das interfaces.

A Figura 5.5 mostra alguns exemplos de concentração de custos de acordo com cada fase.

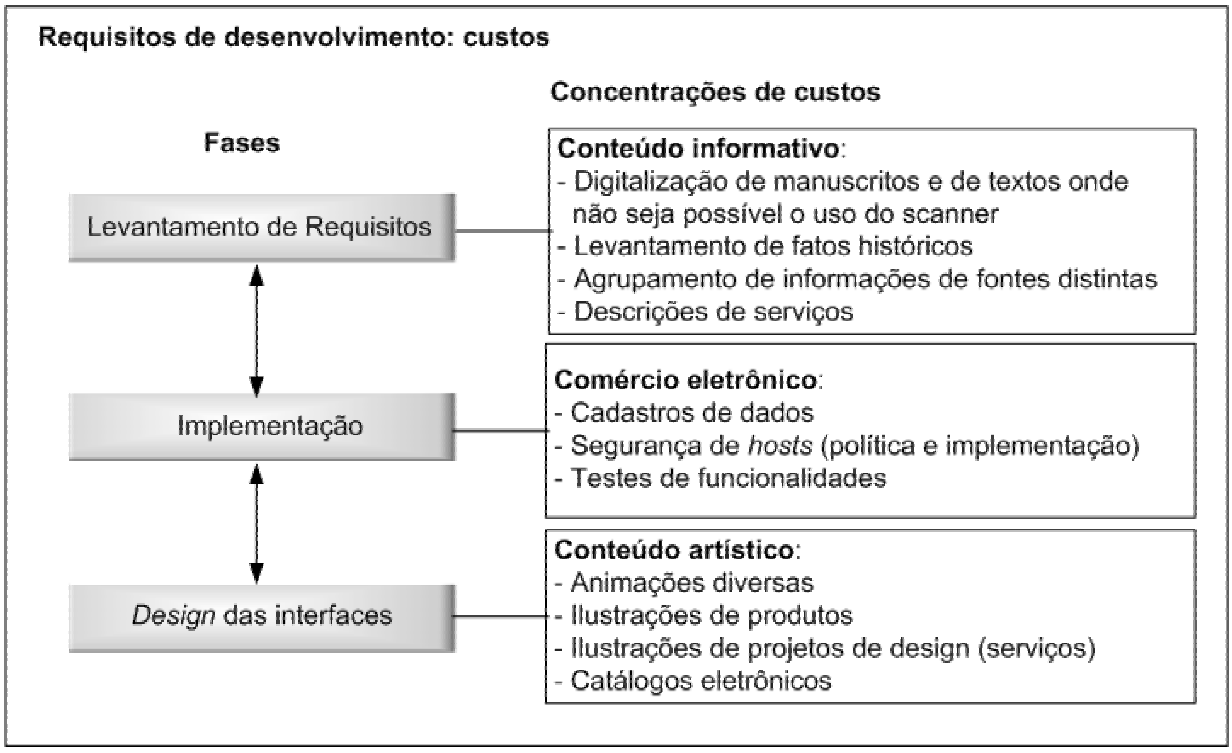


FIGURA 5.5 - Concentração de custos.

5.1.4 - Modelagem do Domínio

Ao analisar os assuntos que fazem parte de um domínio já se podem estabelecer as perspectivas de desenvolvimento de interfaces. O próximo passo para entender o sistema é fazer a modelagem do mesmo. Os recursos da UML utilizados na fase de levantamento de requisitos representam visões estáticas e dinâmicas que ilustram as características do projeto e do sistema (*Web site*). As características ou necessidades do projeto são mostradas através dos diagramas de atividades e as necessidades do sistema são mostradas através dos diagramas de casos de uso.

5.1.4.1 - Interações e Atividades

Os *Web sites* são sistemas que normalmente têm interação com três tipos de atores: o cliente que é o "proprietário" do domínio, a equipe de desenvolvimento e administração e os usuários que são usuários da entidade ou da empresa e os internautas. O cliente e a equipe interagem com o sistema com o objetivo de desenvolvê-lo e administrá-lo para que os usuários possam fazer uso.

Com os diagramas de atividades são mostrados os fluxos das atividades, ou fluxos de trabalho, que envolvem o projeto como, por exemplo, as atividades do cliente, da equipe e dos usuários mostrando etapas seqüenciais.

A Figura 5.6 mostra um diagrama de atividades com os principais atores, atividades e interações de um *Web site*.

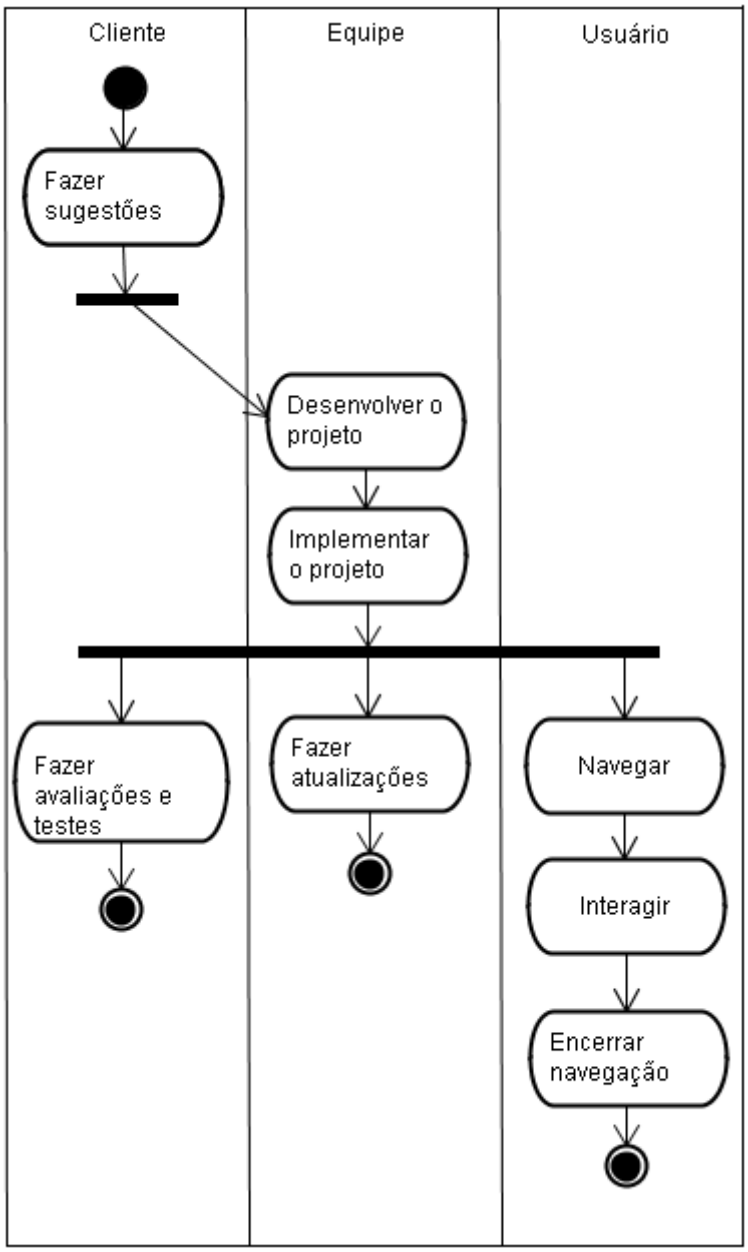


FIGURA 5.6 - Principais atividades e interações de um *Web site*.

- **Cliente**: o cliente de um *Web site* representa o proprietário ou o responsável por um domínio também chamado de contato da entidade. O cliente normalmente é quem

sugere **o que** precisa ser feito e como gostaria que fosse o *design* e o funcionamento geral do *Web site*, sem preocupações de **como** será feito. No entanto, o papel desempenhado pelo cliente ilustra atividades do projeto e não diretamente do sistema.

A Figura 5.7 mostra um diagrama de atividade com as ações que um cliente desempenha em um projeto de *Web site*.

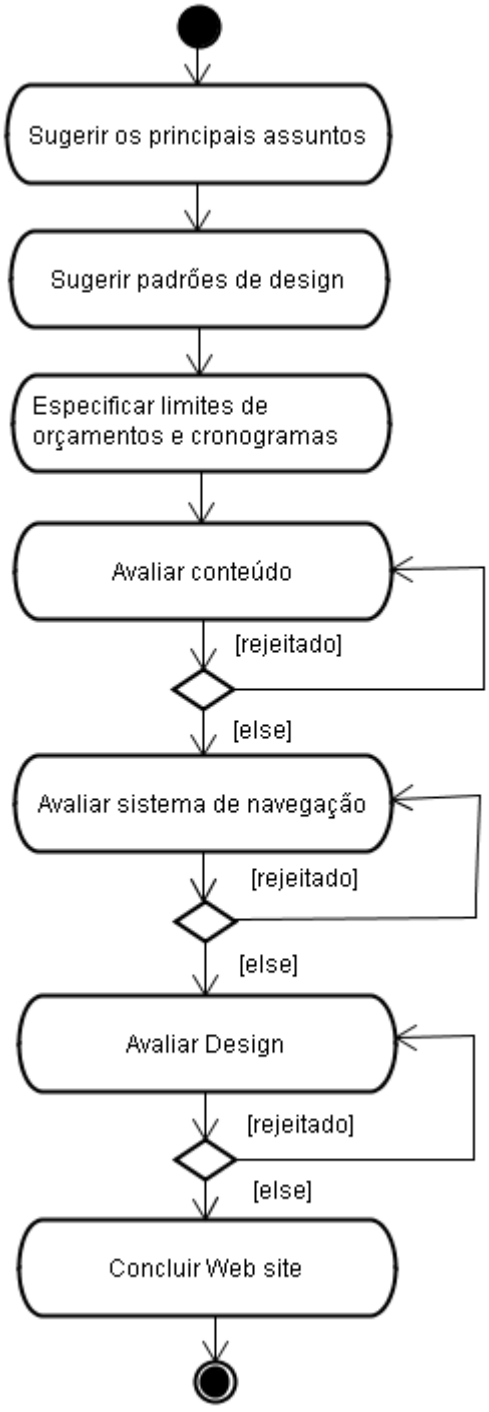


FIGURA 5.7 - Diagrama de atividades com as ações do cliente.

Na figura 5.7 utiliza-se uma conveniência que é a palavra reservada *else* para marcar uma transição de saída, representando o caminho a ser seguido, caso nenhuma outra expressão de proteção seja avaliada como verdadeira (Booch, 2000).

- **Equipe de desenvolvimento e administração**: interage de forma indireta com o sistema e de forma direta com o projeto, podendo fazer o desenvolvimento de novas interfaces, atualizações, acréscimos e reduções de conteúdos em interfaces desenvolvidas anteriormente; fazendo estudos de viabilidades tecnológicas, bem como a migração de tecnologias, quando necessário. A equipe desempenha atividades conforme instruções do cliente com o objetivo de disponibilizar ao usuário possibilidades de navegação e interação com o sistema desenvolvido.

  A Figura 5.8 mostra um diagrama de atividades com as principais ações da equipe de desenvolvimento e administração para com o *Web site*.

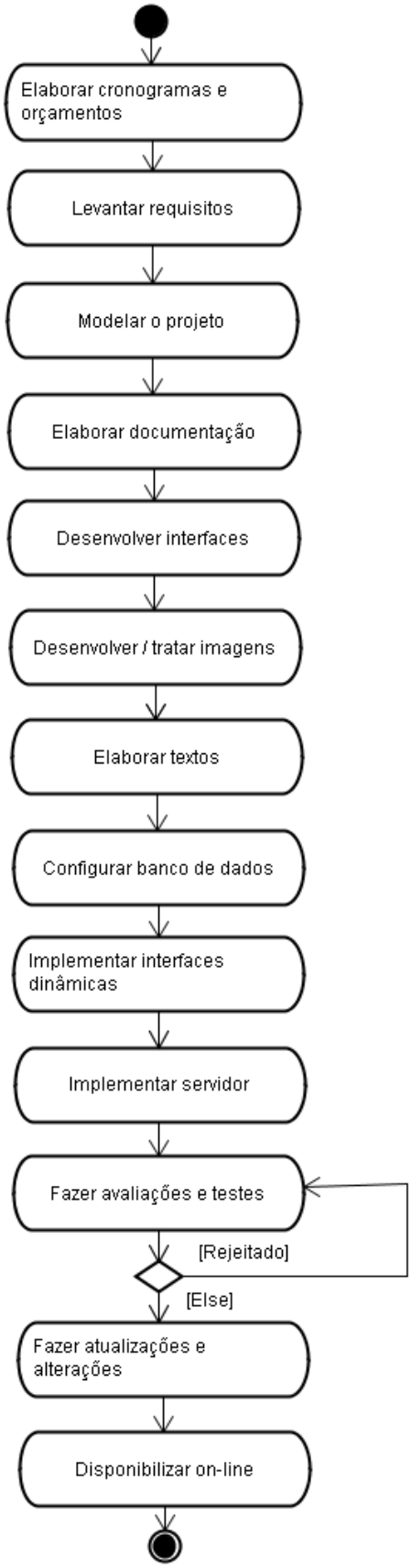


FIGURA 5.8 - Diagrama de atividades com as ações da equipe.

- **Usuários**: os usuários são as pessoas que fazem uso do sistema podendo ser fornecedores de dados e informações ou pessoas que fazem uso do sistema na forma de navegação e interação. Os usuários de um sistema podem ser os usuários da entidade ou os usuários da *Web*, chamados também de internautas.

  - Usuários da entidade: os usuários da entidade realizam atividades como, cadastros de produtos, atualizações de estoque e preços etc. Os usuários da entidade desempenham algum papel quando o *Web site* dispõe de interfaces dinâmicas, principalmente em *Web sites* de comércio eletrônico onde a equipe desenvolve o sistema para os internautas e para usuários da entidade que fazem lançamentos e atualização de dados.

  - Internautas: os internautas são os usuários principais do *Web sites* pois fazem uso do sistema independente do tipo de interface existente no mesmo. Quando o sistema dispõe somente de interfaces estáticas os usuários podem navegar de uma interface para a outra e podem interagir com o sistema quando este dispõe de interfaces dinâmicas.

  A Figura 5.9 mostra um diagrama de caso de uso com as principais ações dos usuários com o *Web site*.

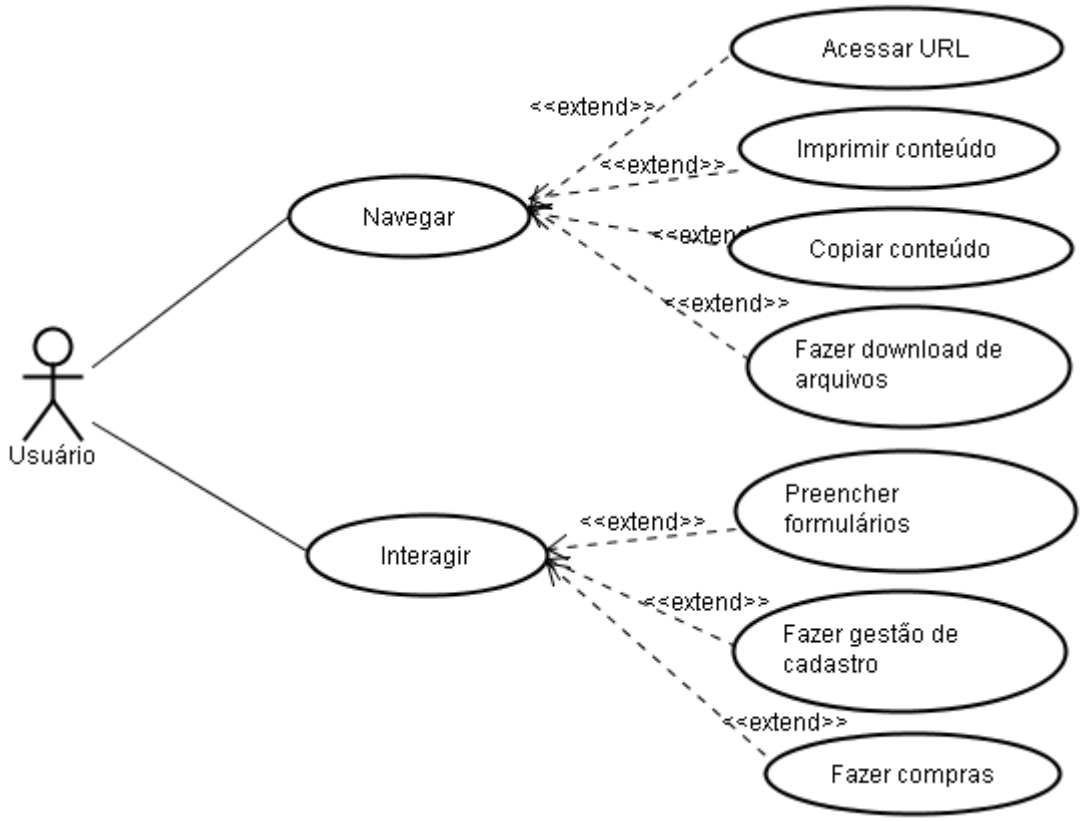


FIGURA 5.9 - Diagrama de casos de uso com as ações dos usuários.

O usuário depende de aprovações da equipe e do cliente para que o *Web site* esteja disponível *on-line*. O usuário interage com o sistema através de um navegador podendo inserir o URL no *location* do navegador, acessar a página e fechar o navegador quando concluir sua navegação.

O usuário acessa um domínio, composto por interfaces que se relacionam com outras interfaces através de algum método. Estas interfaces possuem atributos do tipo *link*, textos, imagens e formulários. Porém, diferente do cliente e da equipe, o usuário vê somente o *design*, o conteúdo, a arte e não a engenharia através da qual o sistema foi desenvolvido. Se uma interface possui somente textos, imagens e *links* o usuário só pode copiar, fazer *download* de arquivos ou escolher outro *link*. Se a interface dispõe de formulários o usuário pode interagir de diversas formas.

Em uma interface do sistema tem-se um atributo de tipo *link* cujo contexto representa uma classe navegacional que faz parte do conteúdo do domínio. Um usuário realiza a operação "Navegar" que é "Clicar em um *link*". O sistema então, apresenta ao usuário o contexto escolhido. Diante deste contexto o usuário poderá realizar operações específicas como

incluir, consultar, alterar e excluir dados (exemplo de um carrinho de compras), ou repetir a operação anterior que é "Clicar em outro *link*". O usuário pode escolher "Clicar em outro *link*" tendo ou não realizado outras operações disponíveis dentro deste contexto.

A Figura 5.10 mostra um exemplo de interfaces estáticas e dinâmicas em um diagrama de classes com os nomes, atributos e métodos conforme o tipo da interface.

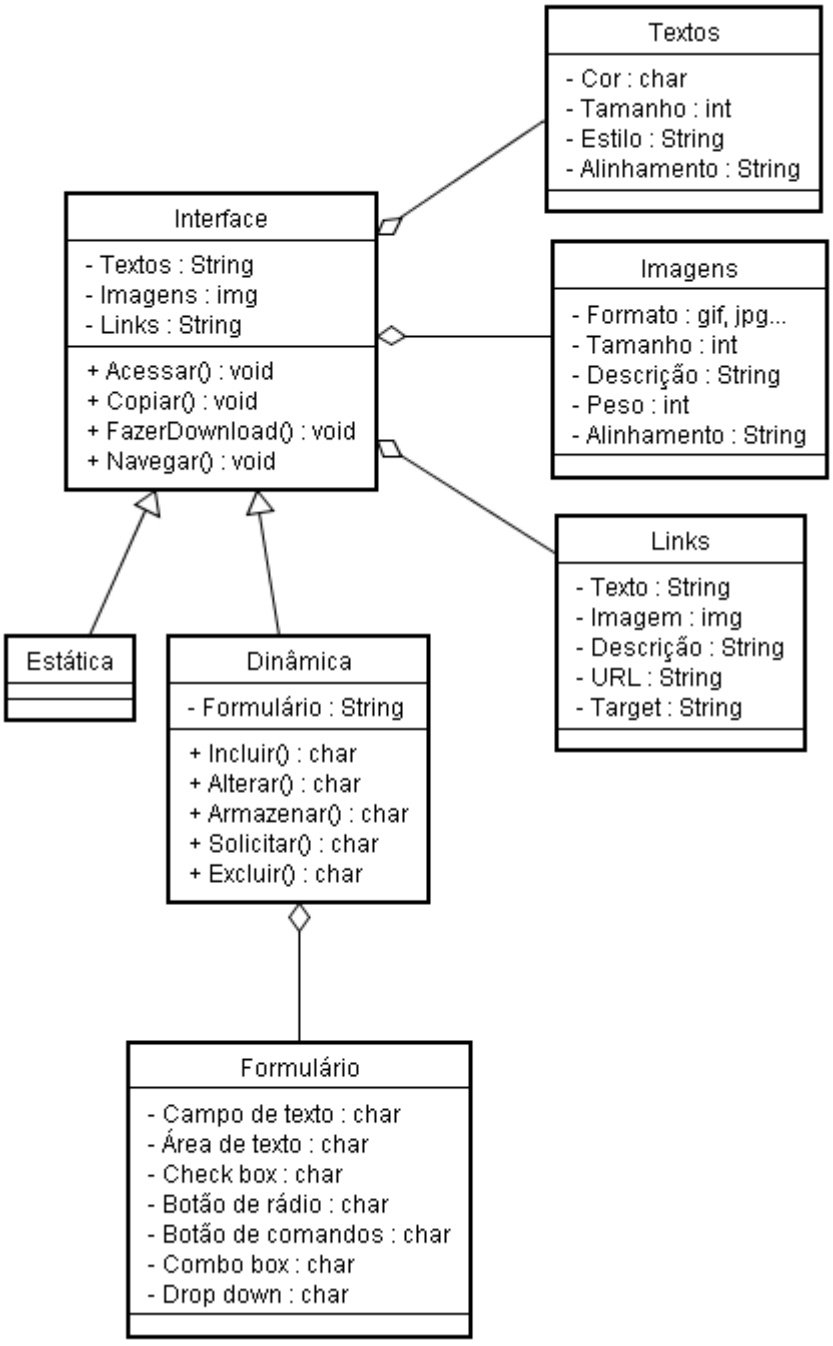


FIGURA 5.10 - Interfaces estáticas e dinâmicas mostradas em um diagrama de classes.

Conforme mostra a Figura 5.10 as interfaces dinâmicas apresentam métodos e atributos que as interfaces estáticas não apresentam. Através de um atributo do tipo formulário começam as interações dos usuários através de métodos que são específicos de interfaces dinâmicas. As operações executadas em interfaces dinâmicas que não tenham início a partir de um formulário são as mesmas executadas em interfaces estáticas.

5.1.4.2 - Modelagem das Interfaces Dinâmicas

A modelagem das interfaces estáticas visa proporcionar aos desenvolvedores *Web* uma visão formalizada do uso de um *Web site*. No entanto, quando um *Web site* tem como parte integrante interfaces dinâmicas ou interfaces que são geradas dinamicamente através de consultas a bancos de dados a modelagem assume características muito semelhantes às utilizadas na engenharia de *software*. A modelagem das interfaces dinâmicas é mostrada através dos diagramas de casos de uso, dos diagramas de classes e dos diagramas de seqüência.

Embora a engenharia de *software* tenha características diferentes da engenharia *Web*, quando há situações de interfaces dinâmicas o sistema de modelagem de ambas seguem os mesmos padrões. Para a modelagem das interfaces dinâmicas, o PDW-UML faz uso da UML como se faz para modelagens de outros sistemas, pois nesse ponto embora os fins sejam diferentes, os meios são muito semelhantes.

A modelagem da interface "produtos" é mostrada a seguir através dos casos de uso "fazer gestão de cadastro" e "fazer compras", onde o ator que interage com o sistema é o usuário.

- **Casos de uso**: um diagrama de caso de uso mostra um conjunto de casos de uso e atores e seus relacionamentos. Estes diagramas são utilizados para ilustrar as necessidades um sistema, observável por um ator. Através dos elementos dos diagramas de casos de uso é possível uma visualização de quem são os atores do sistema e quais os tipos de interação que o sistema permite (Booch, 2000).

  A Figura 5.11 mostra um diagrama de casos de uso com os principais tipos de necessidades de interfaces dinâmicas.

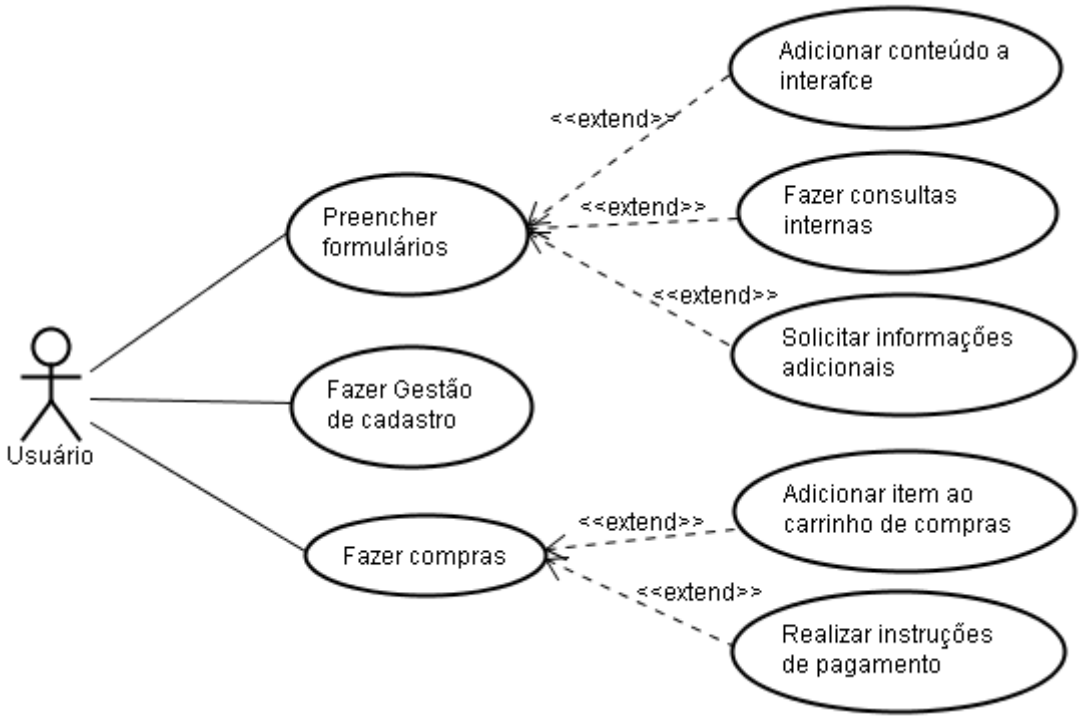


FIGURA 5.11 - Casos de uso de interfaces dinâmicas.

O caso de uso mostrado na Figura 5.11 mostra necessidades de uso de um sistema que tenha interfaces do tipo formulário, contextual e interativa (estereótipos na Tabela 5.3).

- **Classes**: Um diagrama de classes mostra um conjunto de classes, interfaces, colaborações e seus relacionamentos. Através de um diagrama de classes representa-se a estrutura de um sistema; a modelagem da visão estática e os serviços que o sistema deverá fornecer aos usuários finais (Booch, 2000).

  A Figura 5.12 mostra um diagrama de classes com um "carrinho de compras", onde há "produtos" e um "item de compra" que armazena uma compra. O diagrama contém o nome das classes, os atributos e os métodos.

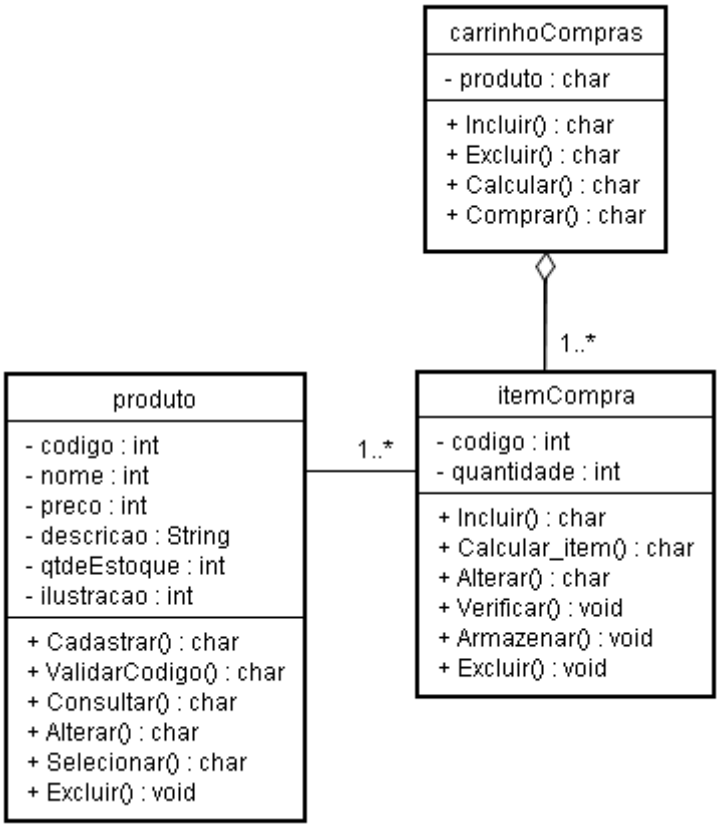


FIGURA 5.12 - Diagrama de classes com o exemplo de um carrinho de compras.

- **Interações**: as interações do sistema são mostradas através dos diagramas de seqüência utilizados na UML para a modelagem dos aspectos dinâmicos do sistema, como a interação de objetos que desempenham uma operação ou parte da funcionalidade do sistema. Como os diagramas de interação podem ser utilizados para fazer a modelagem de um determinado fluxo de controle de um caso de uso, o PDW-UML faz uso dos diagramas de seqüência para mostrar períodos durante os quais os objetos desempenham as suas ações.

  A Figura 5.13 mostra um diagrama de seqüências do caso de uso "Gestão de cadastro", considerando ações de um usuário da entidade que:

  1) Cadastra um produto, cria um código para este produto, verifica se este produto ou este código já não existe no sistema e obtendo as condições para a realização retorna uma mensagem de confirmação de realização do cadastro.

  2) Entra com os dados referentes ao produto, armazena estes dados e obtém uma mensagem de que os dados foram armazenados.

3) Faz alteração dos dados referentes ao produto informando o código do produto, verifica os dados do produto antes de alterar, confirma a alteração e recebe uma mensagem de confirmação de alteração.

4) Entra com os dados de alteração, armazena os dados no sistema e recebe uma confirmação de atualização de dados.

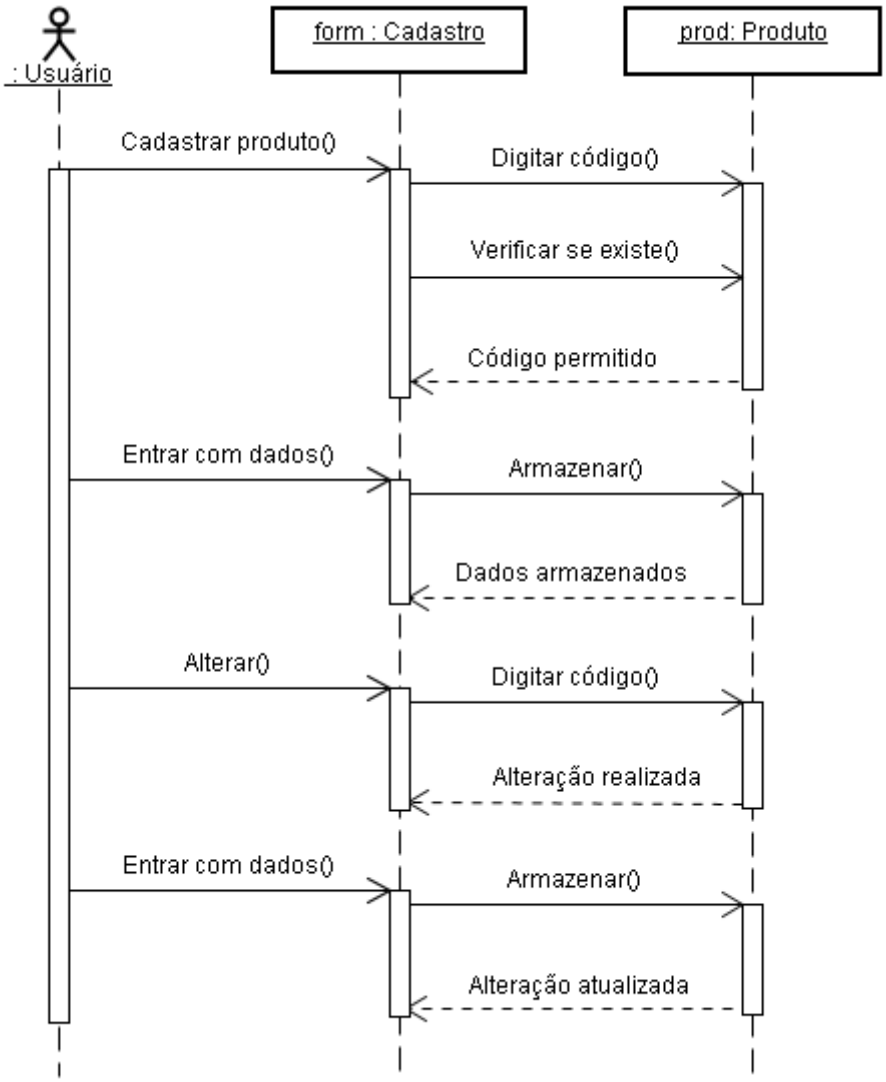


FIGURA 5.13 - Diagrama de seqüências do caso de uso Gestão de cadastro.

Quando um produto já está cadastrado e disponível o usuário (internauta) faz uso do sistema realizando compras *on-line*.

A Figura 5.14 mostra um diagrama de seqüências do caso de uso "Fazer Compras" onde o usuário interage com o sistema fazendo consultas, inclusões de produtos ao

carrinho de compras, fazendo cálculos e demais procedimentos de compra como mostrados a seguir.

1) Um usuário faz uma consulta *on-line* para que através da sua consulta possam ser listadas as categorias de produtos disponíveis. O usuário faz uma consulta de acordo com uma seleção e recebe um retorno com a lista dos produtos.

2) Selecionar um produto para incluir em uma classe chamada itemCompra e retorna uma confirmação de armazenamento do produto.

3) Calcular o valor do produto solicitando ao sistema que realize o cálculo, em seguida verifica-se o valor obtido e retorna uma mensagem de realização do cálculo.

4) Alterar produtos armazenados no item de compras excluindo produtos e/ou adicionando outros, verificar os produtos atuais e retorna uma mensagem com a lista dos produtos armazenados.

4) Entrar com dados de alteração sobre os produtos como quantidade, região, opções de entrega etc. e retorna uma mensagem de confirmação da alteração.

5) Concluir as compras conforme os produtos e suas respectivas informações conforme se encontram no item de compras, realizar o cálculo final e retorna a confirmação da compra.

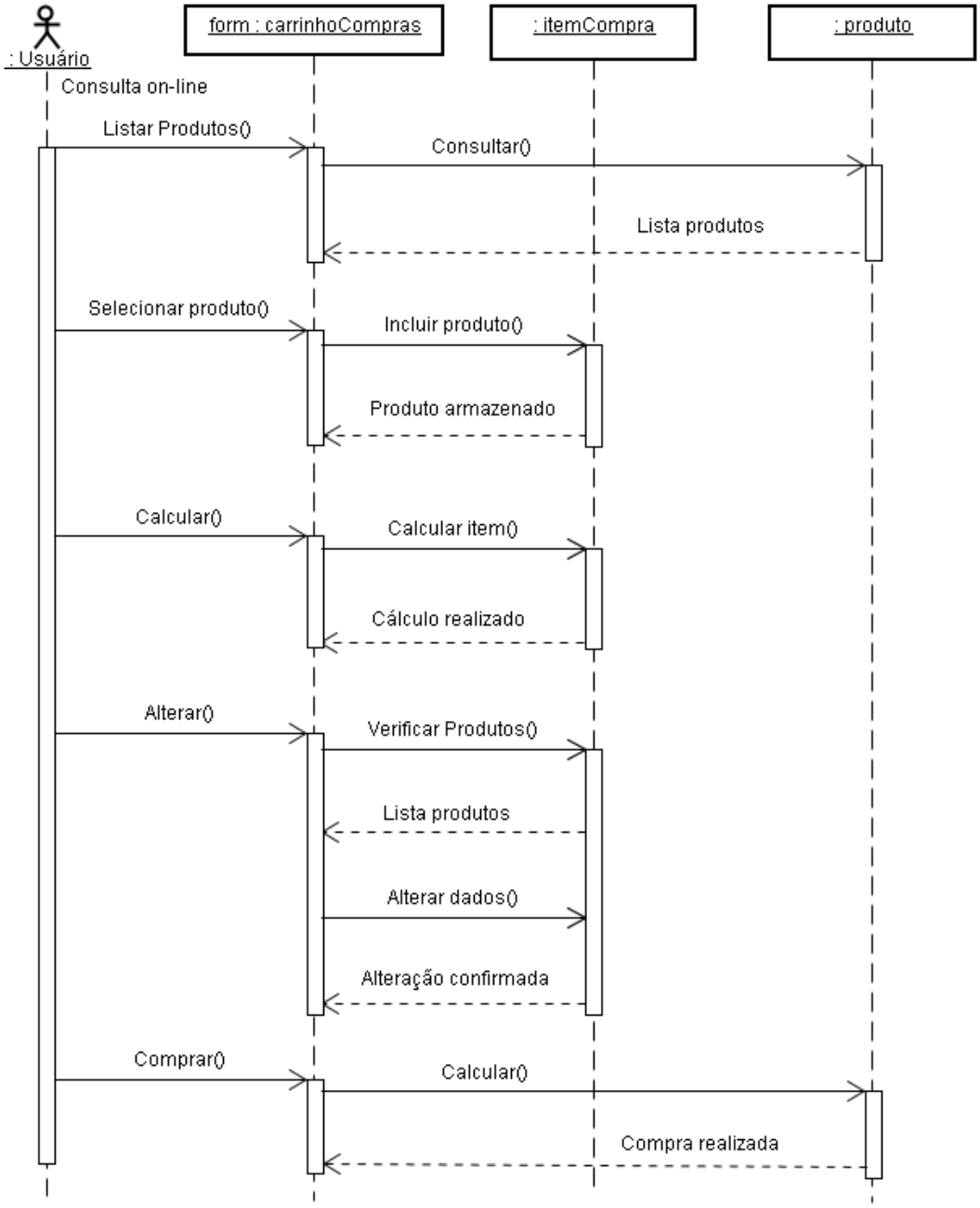


FIGURA 5.14 - Diagrama de seqüências do caso de uso Fazer Compras.

A utilização dos diagramas de interação como diagramas de seqüência tem o objetivo de mostrar o aspecto do modelo que enfatiza as interações de objetos, organizadas em uma seqüência de tempo e de mensagens trocadas. Os diagramas de seqüência permitem uma ênfase maior aos quesitos de seqüência, tornando mais fácil de ser visualizada a ordem na qual as coisas acontecem, o que facilita a visualização das possíveis seqüências dos usuários *on-line* (Booch, 2000).

5.1.4.3 - Componentes

Através do diagrama de componentes, mostra-se a implementação física do sistema expondo as relações entre seus componentes e a organização de seus módulos durante a sua execução. O diagrama de componentes descreve os componentes de *software* e suas dependências entre si, representando a estrutura do código gerado. Os componentes são a implementação na arquitetura física dos conceitos e da funcionalidade definidos na arquitetura lógica (classes, objetos e seus relacionamentos) (Booch, 2000).

A Figura 5.15 mostra um diagrama de componentes com os arquivos que são utilizados para compor o código fonte de uma interface com estrutura dinâmica, gerada dinamicamente a partir de uma consulta a um banco de dados.

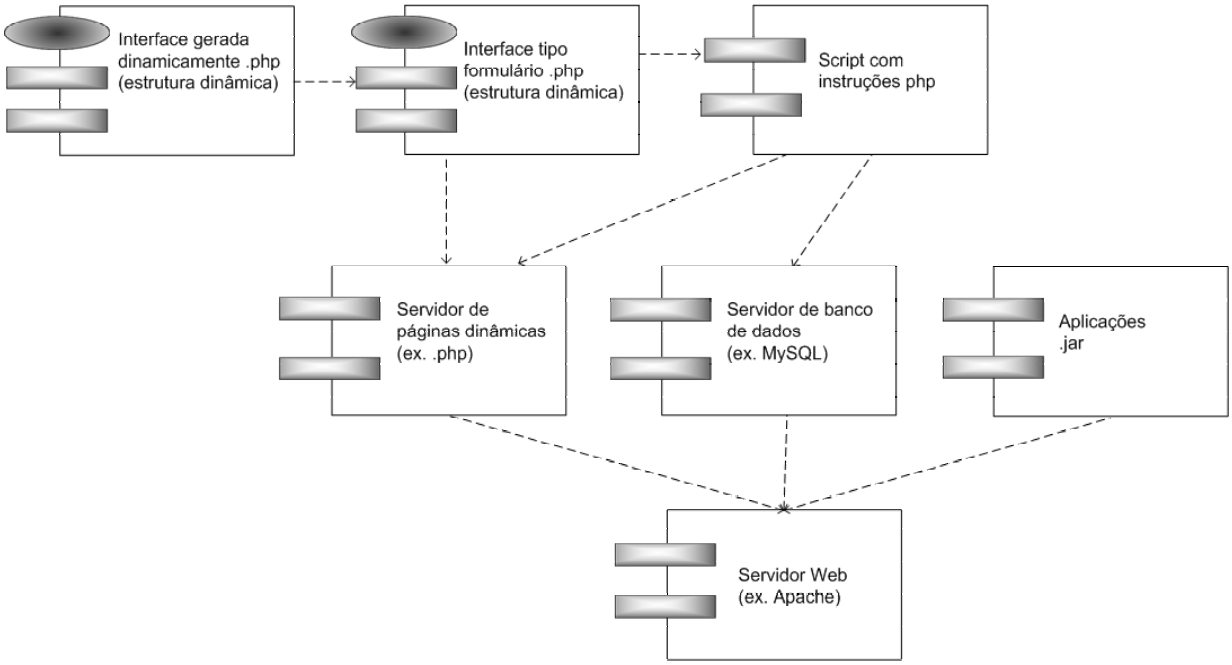


FIGURA 5.15 - Diagrama de componentes.

A modelagem proposta pelo PDW-UML para um sistema de *Web sites* com interfaces estáticas e dinâmicas utiliza exemplos básicos da UML para permitir uma linguagem comum aos desenvolvedores e para auxiliar no processo de implementação, conforme proposto na Seção 5.2.

5.2 - Fase 2: Implementação

Nesta fase faz-se a análise e a seleção dos requisitos obtidos na primeira fase e utilizam-se os requisitos selecionados para compor os conteúdos que formam as interfaces; elabora-se a

documentação; definem-se as estruturas; planejam-se os níveis de URLs; elaboram-se os textos e as imagens; desenvolvem-se os arquivos e diretórios e fazem-se os testes de navegação a partir da máquina servidora.

5.2.1 - Documentação

Antes de iniciar um *Web site* o desenvolvedor precisa de um projeto para saber o que será feito, como será feito, que nomes serão atribuídos aos arquivos e diretórios para que o sistema de desenvolvimento e administração possa ser entendido com mais facilidade. Mas como ocorre na maioria dos *softwares* a documentação faz parte de todas as fases, desde análise dos requisitos até a conclusão do *Web site*.

Na documentação é onde se registram as informações referentes ao *Web site*. Neste trabalho a documentação está sendo mostrada na segunda fase que é a fase onde há necessidades predominantes de registros de informações. No entanto, a necessidade de registrar informações começa durante o levantamento de requisitos, e se estende por toda a fase de implementação, *design* e administração, conforme a seqüência em que o sistema seja desenvolvido. A seguir são mostrados alguns itens considerados necessários para o modelo de documentação proposto pelo PDW-UML.

5.2.1.1 - Descrição do Domínio

A descrição do domínio é uma forma de registro de informações sobre o que o domínio representa e quais são as divisões e subdivisões. Para facilitar o levantamento de requisitos é necessário que se conheçam os objetivos do domínio, que é a empresa, o produto, o serviço ou as informações para os quais o *Web site* está sendo desenvolvido, quais serão os principais *links* e suas respectivas interfaces etc.

5.2.1.2 - Informações Gerais

As informações gerais representam os registros dos recursos utilizados em um sistema e informações diversas de interesse de pessoas envolvidas direta e indiretamente em um projeto como, por exemplo, recursos de *software* e *hardware* necessários, pessoas envolvidas, cronogramas, orçamentos etc.

Estas informações além de orientar o próprio projeto podem servir de parâmetro para outros projetos como, por exemplo, fornecendo dados do número de pessoas envolvidas, prazos, custos.

A Tabela 5.1 mostra uma relação de informações gerais referentes a um projeto de *Web site*.

TABELA 5.1 - Informações gerais de um projeto de *Web site*.

Domínio	www.empresa.com.br
Versão	1.0, inicio dia/mês/ano versão atual + data
Pessoas envolvidas	Nome do *Web master*: Nome do administrador: Nome do contato da entidade
Colaboradores	Colaborador 1: Colaborador 2:
Softwares utilizados	Documentação: Pacote Office, Adobe Writer. Codificação: Bloco Notas, php editor. Visualização: Navegadores Diversos. Edição de imagens: Macromedia Fireworks, Freehand. Animações: Macromedia Flash. Servidor de páginas dinâmicas: IIS, Apache Transferência de arquivos: SSHSecure ShellClient, Putty, WS-FTP.
Extensões de arquivos	html , txt, php, gif, jpg, jpeg.
Linguagens	HTML, PHP, CSS, JavaScript.
Orçamentos	Custos estimados por página, criações de imagens, hora técnica.
Cronograma	Previsão de início, provisão de entrega da fase um, previsão de entrega da fase dois, previsão de entrega da fase três e previsão de entrega do projeto final após a realização de todos os testes.

5.2.1.3 - Convenções

Algumas convenções registradas na documentação de um projeto podem colaborar para que a padronização seja mantida, mesmo quando diferentes profissionais administrem um mesmo domínio. Tanto a parte visual como a codificação, quando seguem algumas convenções tendem a facilitar o trabalho de desenvolvedores e administradores. Alguns exemplos de convenções são mostrados a seguir.

- **Nomenclatura de arquivos**: um exemplo de convenção para arquivos e diretórios é determinar que os nomes sejam mnemônicos para que os nomes dos arquivos

possam ser associados aos seus respectivos conteúdos e os nomes dos diretórios aos respectivos *links*.

Uma forma de convenção para caracteres é que sejam usados, para nomes de arquivos e diretórios, somente letras minúsculas, tanto para nome como para extensão, seja arquivo de imagem ou de conteúdo; que não seja utilizado nenhum tipo de caractere (ç, acentos, traços, barras) que possam não ser reconhecidos por um *host* ou que possam levar o usuário a ter dúvida quanto ao nome do arquivo ou dependendo da configuração do teclado, não tenha a opção daquele caractere.

- **Abertura de janelas**: algumas convenções de abertura de janela que podem ajudar na padronização das interfaces são mostradas a seguir.

  - Abertura na própria janela: instruções para que os *links* abram na própria janela (substituindo a página atual) quando a página chamada fizer parte do mesmo domínio.

  - Abertura em janela independente: instruções para que os *links* abram em uma janela independente quando a página chamada fizer parte de um outro domínio.

  - Abertura em *pop-up*: instruções para que os *links* sejam abertos em janelas *pop-up* quando o conteúdo a ser exibido não requeira uma nova página e nem seja conveniente a abertura na própria janela. Ex. *link* para vídeo, imagens de alta resolução, informações de datas de eventos e de informações temporárias.

- **Utilização de cores**: existem várias formas de se fazer convenção do uso de cores. Estas convenções podem ser apenas informações ou justificativas que mostrem que as cores que estão sendo usadas não são por acaso, mas por coerência e familiaridade com motivos como: os motivos originais da empresa; com base no logotipo a partir do qual adotam-se também as cores para compor as interfaces e cores extraídas da tabela *Web safe* que traz as 216 cores seguras para a *Web*. Como as cores usadas nas páginas poderão ser necessárias também para outros *softwares*, como editores de imagens e *software* de animação, por exemplo, registram-se todos os dados referentes à cor. O nome da cor é utilizado mais para uma referência ou linguagem comum entre os administradores, pois o que mais se utiliza em arquivos de códigos é o código hexadecimal e em aplicativos gráficos o que mais se usa é o RGB (*red, green and blue*).

A Tabela 5.2 mostra o nome de algumas cores, a ilustração, o RGB e o código hexadecimal que compõem a Tabela *Web safe colors*.

TABELA 5.2 - Exemplos de referências de cores.

Nome	**Cor**	**RGB**	**Hexadecimal**
Black (Preto)		R:0 G:0 B:0	#000000
White (Branco)		R:255 G:255 B:255	#FFFFFF
Blue (Azul)		R:0 G:0 B:255	#0000FF
Red (vermelho)		R:255 G:0 B:0	#FF0000
Green (verde)		R:0 G:255 B:0	#00FF00

- **Exposição de textos**

  - **Estilo** (Itálico/Negrito/Sublinhado): o sublinhado quando usado em textos que não são *links* podem confundir o usuário uma vez que é o padrão dos *links*. Assim, uma forma de convenção para os *links* é que sejam sempre sublinhados e os demais textos não o sejam.

  O uso de negrito no meio de um bloco de texto pode confundir o usuário que normalmente encontra títulos e subtítulos em negrito. Uma convenção para o negrito é que seja utilizado somente para títulos e subtítulos. Os textos que mereçam destaque dentro de um bloco podem ter seu destaque através de parágrafos identados e aspas. O itálico, mais utilizado para destacar palavras de outro idioma, também pode ser uma convenção de destaque de texto, desde que não seja texto muito longo, pois pode dificultar a leitura, dependendo das condições do vídeo do usuário.

  - **Fonte** (*face e size*): como nem todos os sistemas operacionais dispõem das mesmas fontes, uma forma de convenção de uso que evita que os textos tenham diferenças muito grandes entre um e outro computador é agrupar fontes semelhantes e que sejam as mais usadas na maioria dos sistemas operacionais. Um exemplo de agrupamento são as fontes "*times*, *arial*, *verdana*, *helvetica*, *sans-serif*" que têm uma certa semelhança de estilo e tamanho. No caso de não existir a primeira mencionada no parâmetro *face* da *tag* <FONT>, o sistema utiliza a que estiver mencionada logo em seguida e assim sucessivamente.

Os tamanhos de fonte podem ser especificados em px (*pixels*), pt (pontos). Em *pixel* os tamanhos são flexíveis e mudam de tamanho conforme a resolução do monitor do usuário. Os pontos são fixos, independentes da resolução e aparecem sempre do mesmo tamanho. Assim, uma convenção para o uso de *pixels* pode fazer com que não haja problemas de um usuário se deparar com fontes muito pequenas nem muito grandes. Os tamanhos mais utilizados são 12px para textos em geral e 14px para títulos e subtítulos. Outros tamanhos são necessidades eventuais como 8px para notas de rodapé e 20px (18, 16) para títulos em forma de *banners*.

- **Alinhamentos** (direita, esquerda, justificado, centralizado): as formas de alinhamento podem ajudar ou atrapalhar a leitura do usuário. Assim, algumas convenções de alinhamentos podem facilitar a leitura e a compreensão de uma interface como, por exemplo: que não haja alinhamentos à direita; que os textos curtos (até duas linhas) sejam alinhados à esquerda e quando houver mais de duas linhas que sejam justificados e; somente os títulos sejam centralizados.

- Maiúsculas/minúsculas: o uso de letras maiúsculas na *Web* representa que o texto está "gritando" para chamar a atenção do usuário. Por isso uma forma de convenção que pode sanar este problema é fazer uso letras maiúsculas somente para os títulos da página (não de textos de conteúdo), a primeira letra após um ponto e para nomes próprios conforme determina a norma culta do idioma.

- **Utilização de peso máximo (Kbytes) por interface**: o peso ideal de cada interface depende do segmento de um *Web site*. Se o perfil predominante for de usuários de banda larga, não há necessidade de reduções extremas para formar o montante do conteúdo de cada interface. Quando o conteúdo é de interesse geral, onde a maioria dos usuários utiliza banda estreita (realidade brasileira, segundo o IBOP *e-Ratings, e* MCT, Juliaz, 2000 e MCT, 2004), primeiro se convenciona um peso máximo por interface para depois elaborá-la. Independente do perfil do usuário de um *Web site* o uso de uma convenção do peso máximo por página facilita a navegação, pois fornece um parâmetro para o usuário. Se uma página tiver um tempo de carregamento muito superior às demais, o usuário pode desistir do acesso considerando que aquele *link* não dispõe de uma interface correspondente.

- **Ordem de disposição dos atributos de uma interface**: algumas convenções que padronizem a disposição dos atributos nas interfaces do *Web site* podem facilitar a leitura e o entendimento global do domínio. Alguns exemplos são:

  - *links*: manter a disposição dos *links* nas interfaces de acordo com as perspectivas de acesso ou importância de um assunto perante os demais. Não havendo perspectivas de acesso nem diferenciação de importância, uma forma de disposição é a ordem alfabética crescente. Havendo uma convenção, elimina-se a situação em que em uma interface alguns atributos como *links*, por exemplo, estejam em uma ordem e em outras interfaces em uma ordem diferente. Uma disposição não padronizada pode levar o usuário a ter dúvidas se já esteve naquela interface ou não.

  - Tópicos em geral: para os tópicos, em geral, pode-se convencionar que sejam distribuídos de acordo com sua importância em relação aos demais que compõem a mesma interface. Não havendo esta distinção pode-se utilizar a ordem alfabética crescente.

  - *Home Page* (HP): existem algumas regras que podem melhorar o *design*, a estética e a funcionalidade de uma HP. No entanto, a HP representa uma interface única dentro de um *Web site*. Quando houver necessidade de convenções estas devem ser especificas para cada projeto em questão, pois o que vale para a HP de um portal pode não valer para a HP de um *Web site* pequeno (composto de poucos assuntos); o que vale para a HP de um *site* comercial pode não valer para um *site* pessoal. Convenções para a HP não serão abordadas neste trabalho.

- **Organização das *tags***: algumas linguagens são sensíveis à caixa (*case sensitive*), outras não. O HTML, por exemplo, não é sensível. Para uma melhor organização e visualização das *tags*, pode-se convencionar que as *tags* HTML e seus respectivos parâmetros sejam com letra maiúscula e os valores atribuídos aos parâmetros fiquem entre aspas e com letras minúsculas. As demais linguagens seguem suas exigências de *case sensitive*. Quando não houver exigências segue-se a convenção de uso (ex. <IMG SRC=”imagem.gif” WIDTH=”20” HEIGHT=”15”>).

5.2.1.4 - Modelagem

O desenvolvimento de um *Web site* não é um processo de uma única vez, mas é um processo que requer atualizações freqüentes durante toda a permanência *on-line*. Em conseqüência

disso nem sempre são os mesmos profissionais os responsáveis por um domínio durante toda a sua existência. Por isso a modelagem de um sistema pode auxiliar profissionais já familiarizados com o domínio e novos profissionais que precisam entendê-lo para realizar seu trabalho com mais coerência.

Conforme seja a complexidade do *Web site*, suas necessidades de geração de páginas dinâmicas, interações com bancos de dados, *plug-ins* etc, pode-se utilizar um sistema de modelagem como se faz em um sistema de *software* convencional. Necessidades como estas é que fazem com que em alguns aspectos a engenharia de *software* sejam muitos semelhantes aos da engenharia *Web*. Embora os fins sejam diferentes, a modelagem de um sistema previsto pela UML, como os casos de uso, os diagramas de classe e diagramas de interação implantação, podem ser usados para ambos os sistemas.

5.2.1.5 - Atualização e Manutenção

Para que as atualizações e manutenções do *Web site* sejam feitas é necessário que conste na documentação um sistema que mostre passo a passo como o *Web site* foi desenvolvido e como fazer alterações, acréscimos e reduções. Isso permite que o administrador, mesmo não familiarizado com o sistema, possa dar seqüência a um trabalho de atualização e de manutenção.

O sistema de ajuda ou de orientação de administração é justamente um dos itens mais complexos de uma documentação, segundo a engenharia *Web*. Isso porque quando um *Web site* é desenvolvido com uma ferramenta específica, quase todo o treinamento acaba sendo voltado ao uso da ferramenta, como por exemplo, o Microsoft Front Page ou o Macromedia Dreamweaver, (que são os mais usados atualmente). O problema no uso de ferramentas de desenvolvimento é que se mudar a versão o sistema de administração precisa ser refeito para que os passos, de acordo com a nova versão, possam ser seguidos.

Quando se usa codificação manual, o processo de atualização de treinamento é bastante reduzido, pois os códigos serão os mesmos, as seqüências serão as mesmas, a menos que mude a tecnologia e uma parte da codificação tenha que ser refeita.

Um sistema de ajuda para o administrador do *Web site* envolve desde atividades básicas como a criação de arquivos e diretórios, como elaborações de *queries* para consulta a banco de dados; elaboração de interfaces dinâmicas que recebem a saída de consultas de banco de

dados; estrutura das interfaces; elaborações de interfaces e parâmetros necessários a cada *tag* etc.

5.2.1.6 - Documentação *On-line*

Quando um *Web site* apresentar situações em que interfaces sejam geradas dinamicamente através da saída de consultas a bancos de dados é necessário que haja uma documentação *on-line* também, além da documentação normal do *Web site*. Conforme sejam os procedimentos do usuário em uma interface de consulta, pode ocorrer que não haja a resposta esperada, por isso é necessária uma documentação (um sistema de *help*) que instrua o usuário sobre como proceder em caso de dúvidas ou dificuldades. Uma documentação *on-line* cabe somente a situações onde há páginas geradas dinamicamente. Em interfaces permanentes o que se recomenda são mapas do *Web site* e os canais de comunicação como *e-mail*, *instant messenger* ou telefone, mas não uma documentação, uma vez que as interfaces devem apresentar uma organização que sugira claramente onde os assuntos podem ser encontrados.

A documentação *on-line* faz parte do *Web site*, tem interfaces próprias e fica no *host* para que possa ser consultada pelo usuário, ao contrário do restante da documentação que deve ser mantida somente na máquina cliente, pois interessa somente aos administradores. Uma documentação *on-line* registrada na documentação do *Web site* permite que sejam consultados os registros de previsões de interfaces geradas dinamicamente, bem com as previsões de erro e seus respectivos tratamentos.

5.2.1.7 - Armazenamento de Arquivos Originais

Na documentação também deve conter informações de onde e como estão armazenados os arquivos originais do *Web site*. Onde quer que os originais sejam armazenados, seja em um diretório dentro do *root* ou em um outro qualquer, este local deve ser informado na documentação, para que possa ser utilizado conforme necessidades.

Um dos problemas de não armazenar os arquivos originais é que alguns assuntos que não tenham sido utilizados em um primeiro momento poderão ser usados em uma outra oportunidade. O uso das imagens a partir de *bitmaps* também é um problema que ocorre quando não se tem o original (normalmente um vetorial que permite que a imagem seja ampliada e reduzida sem que haja perda da qualidade como as extensões .tif, .png etc.) a partir do qual se possam fazer alterações e compactações.

Independente do espaço que uma imagem possa ocupar em uma interface, quando compactada a partir de um original, é possível obter um índice elevado de compactação com pouca perda de qualidade.

5.2.2 - Análise e Seleção dos Requisitos de Conteúdo

Como conseqüência do volume e da complexidade do que se determina como parte integrante de um sistema, ocorrem variações orçamentárias e no cronograma de um projeto. Por isso a fase de análise e seleção de requisitos exige cuidados e precauções para que não haja supressões ou excessos que possam comprometer a coerência e a objetividade necessárias ao contexto geral do domínio.

Diante de alguns contextos há questões que podem ajudar a reduzir desperdícios e elevar a objetividade seja de um tópico específico, de um assunto composto por vários tópicos ou de uma interface composta por vários assuntos. Uma forma de seleção é considerar assuntos específicos e analisá-los em busca de respostas que possam se adequar às necessidades de um domínio.

Com o resultado da seleção dos assuntos passa-se então, para a análise de perspectivas de elaboração de *links* e suas respectivas interfaces. A seguir são mostradas algumas questões que podem servir de orientação para o desenvolvedor quanto à utilização ou não de um assunto e como este pode ser utilizado na composição de uma interface.

- **Sobre um assunto**: este assunto é importante para o domínio? Precisa ser usado neste momento? É suficiente para que se desenvolva uma interface específica para este assunto? É importante o suficiente para que haja um *link* para este assunto na barra de navegação da *home page*? É importante, mas requer complemento prévio sendo mais adequado que seja chamado em um *link* em um segundo nível? Precisa de complementos? Quais os tipos de complementos que existem nos requisitos do domínio? O assunto precisa ser segmentado? Precisa ser resumido? Precisa ser agrupado a outros assuntos?

  Um assunto que faz parte dos requisitos de um domínio exerce a função de um "papel" de realizar (compor) uma interface. Um assunto selecionado poderá compor uma interface como um todo; mais de uma interface e/ou fará parte de uma interface composta também de outros assuntos. Quando um assunto não é suficiente para

compor uma interface ele pode cumprir seu papel de realizador de forma parcial se tornando um componente de uma outra interface junto a outros assuntos ou é descartado, permanecendo armazenado nos requisitos do domínio, sem cumprir seu papel de realizador de interface.

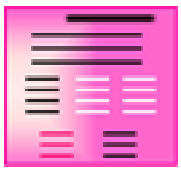
Nem sempre todo o conteúdo referente a um assunto é utilizado em uma interface. Há situações em que parte de um assunto é utilizado e parte é armazenado. Isso acorre quando a parte do assunto que é armazenado tem seu sentido de existência de forma sazonal (datas comemorativas, produtos e serviços típicos de uma estação do ano); os custos de produção de uma interface são muito elevados (vídeos, áudios, imagens) ou exige-se um elevado dispêndio de tempo para sua realização (implantação de recursos tecnológicos sofisticados, trabalhos artísticos que requerem muitas horas de trabalho e serviços especializados).

- **Sobre uma interface**: há conteúdo suficiente para que uma interface seja desenvolvida? Com qual outro assunto do domínio que o assunto em questão se relaciona mais diretamente? Poderiam então, mais de um assunto fazer parte de uma mesma interface? Existe a necessidade de segmentação ou de agrupamentos? Quantas interfaces são necessárias para um mesmo assunto?

  A partir da análise e seleção dos assuntos já se pode estabelecer uma hierarquia de navegação e fazer classificações dos tipos de interface que irão compor o domínio. Os tipos mais comuns de interfaces *Web* são a *home page* que é a interface inicial do *Web site*; interfaces com o "assunto" propriamente dito; interfaces de formulários; interfaces contextuais (geradas dinamicamente); interfaces interativas; interfaces de resumo que podem ser somente com pequenos blocos de textos ou com *links* e resumos de assuntos (intermediários) que serão chamados pelos *links* e as barras de navegação também chamadas de menu, as quais normalmente são parte integrante da *home page* e dos demais tipos de interface.

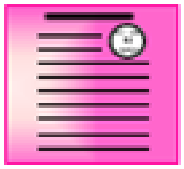
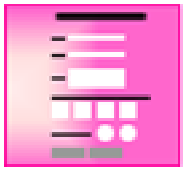
  A Tabela 5.3 mostra um estereótipo e uma breve descrição dos principais tipos de interfaces de *Web site* proposta pelo PDW-UML.

TABELA 5.3 - Estereótipos e descrição de interfaces.

Estereótipo	Descrição
Barra de Navegação	As barras de navegação, também chamadas de menu, são interfaces que trazem os *links* (pontos clicáveis) que levam os usuários às páginas escolhidas. As barras de navegação têm as mais diversas formas, podendo ser verticais, horizontais, com formato oval etc. Podem aparecer no topo, na base, à direita, à esquerda, ao centro, sendo mais comum aparecerem à esquerda com início ao topo. Existem barras de navegação que são a própria *home page* a partir da qual ocorre o desdobramento do conteúdo geral.
Home Page	A *home page* é a interface inicial de um *Web site*. O termo *home page* é uma metáfora do inglês para expressar "página de casa" com a idéia literal de **sala da casa**, onde a *Web* representa o sistema habitacional, o *site* representa um endereço específico dentro do sistema habitacional e a *home* representa o espaço "representativo" de determinado endereço (URL); o espaço onde se recebem visitas, que está "sempre" bem organizado/arrumado para passar uma boa impressão do restante da casa ainda que nem todos os compartimentos sejam visitados. A *home page* é considerada a "porta de entrada" ou "cartão de visitas" do *Web site*. Na prática, a *home page* é a página inicial, mas nem sempre exerce o papel de porta de entrada, pois o usuário pode acessar um domínio a partir de qualquer *link*. Normalmente a *home page* aponta (dispõe *links*) para informações de primeiro nível (em interfaces de segundo nível); passa uma noção geral do contexto do domínio deixando cada assunto específico para sua respectiva interface.
Resumo 	As interfaces de resumos exibem pequenos blocos de textos que poderão ou não, terem uma seqüência. Esses resumos são feitos baseados em alguns métodos de redução como: **Agregação**: que consiste no agrupamento de vários assuntos que representam um único tema em uma interface. **Sumarização**: que é uma forma de representar assuntos extensos com poucas palavras, normalmente seguido de um *link* que sugere o assunto completo. **Omissão**: que é uma amostra do início de um assunto

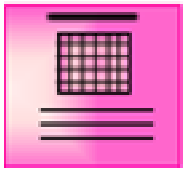
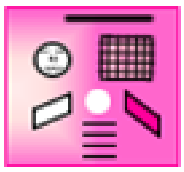
(Continua)

Tabela 5.3 - Continuação

	seguido por um *link* com frases sugestivas como, por exemplo, mais... seqüência... ler artigo... etc...
Conteúdo 	As interfaces de conteúdo podem estar em vários níveis, sendo mais comum em segundo ou terceiro. **Primeiro nível**: quando a interface apresenta um conteúdo específico logo no primeiro nível então se tem o conteúdo na própria *home page*. Isso ocorre normalmente em *Web site* cujo objetivo é dispor ao usuário um único assunto. Quando há *links* existentes nas interfaces de primeiro nível são *links* para tópicos da própria página ou para interfaces de níveis inferiores. **Segundo nível**: as interfaces de segundo nível aparecem normalmente a partir da *home page*. Há situações em que uma interface encerra a sua seqüência logo no segundo nível. Há situações em que é necessário que se desenvolvam mais um nível para que um assunto possa ser completado (as interfaces de segundo nível, normalmente trazem os assuntos de primeiro nível). **Terceiro nível**: as interfaces de terceiro nível vêm normalmente como seqüência de interfaces resumos de um segundo nível. Quando em um segundo nível tem-se uma interface resumo de sumarização ou de omissão, conseqüentemente existe uma interface de terceiro nível para dar seqüência ao que foi sugerido na interface anterior. Há ainda situações em que são necessários mais níveis para a exposição completa de um assunto. Como as interfaces muito extensas não são muito recomendadas, divide-se um assunto formando níveis inferiores mostrando uma parte do assunto e partir de um determinado número de linhas insere-se de um *link* sugerindo continuidade...
Formulário 	As interfaces tipo formulário representam a interface a partir da qual se têm os conteúdos gerados dinamicamente ou os conteúdos contextuais que são gerados conforme os dados de uma consulta. A necessidade das interfaces formulário ocorre quando se tem um volume grande de dados, seja em arquivos texto ou em banco de dados, onde cada usuário poderá ter interesse em informações diferentes, não sendo conveniente manter todos expostos diretamente na interface.

(Continua)

Tabela 5.3 - Conclusão

Contextual 	As interfaces contextuais são as respostas obtidas de consultas feitas a partir de interfaces de formulários, normalmente com interatividades com banco de dados. Chamam-se contextuais, porque por não estão disponíveis permanentemente em um *Web site*. Quando um usuário interage com uma interface tipo formulário (busca ou envio de dados) elaboram-se as interfaces contextuais que exibem um resultado para aquele contexto ou para aquela interatividade específica. Se o usuário acessar uma outra interface, aquele contexto deixa de existir. Novas interfaces poderão ser geradas quantas vezes forem necessárias, desde que haja uma intenção da geração da mesma por parte do usuário.
Interativa 	As páginas interativas têm uma interface permanente, porém o conteúdo das mesmas é construído gradativamente a partir da interatividade de usuários com páginas tipo formulário e de instruções em códigos como tempo de expiração. Este tipo de interface é vista em murais de recado, livros de visitas, oferta/procura de serviços onde as mensagens inseridas pelo usuário são acrescidas a uma interfaces já existente. Estas mensagens poderão ficar na interface por tempo indeterminado ou poderão expirar depois de algum tempo conforme haja instruções no código fonte. Conforme instruções de código novas interfaces podem ser geradas a partir de um número específico de registros, de tempos em tempos, por categoria de assuntos etc. As interfaces interativas diferem-se das contextuais porque as contextuais só existem no período de tempo em que o usuário mantém o acesso àquela interface. As interfaces interativas existem em um *Web site*, porém têm seu conteúdo é alterado, conforme instruções de código e interatividades dos usuários.

As interfaces *Web* são compostas por alguns atributos que são comuns a qualquer tipo de interface como, por exemplo, texto, imagem e *link* e outros que são específicos de interfaces dinâmicas, como é o caso dos formulários. Interfaces do tipo contextual e interativa representam interfaces dinâmicas, mas seus atributos, na maioria das vezes são os mesmos que existem nas interfaces estáticas.

A Figura 5.16 mostra um diagrama com os estereótipos e os atributos que compõem as interfaces.

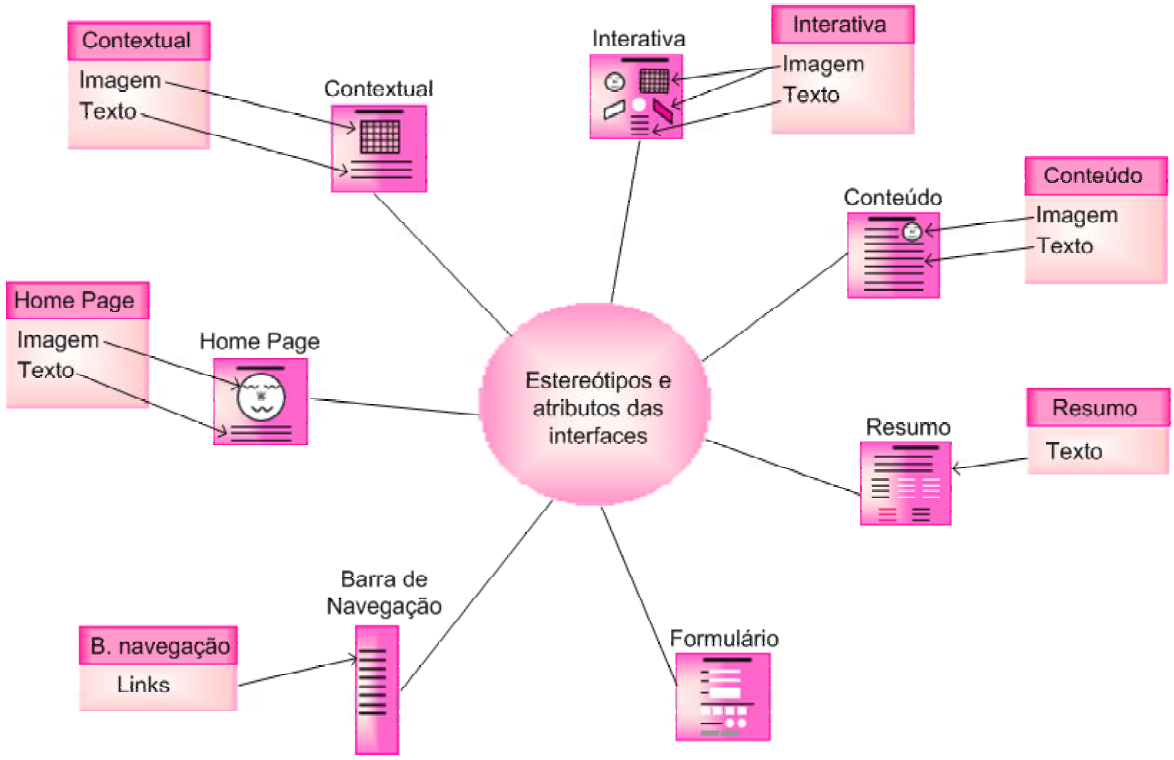


FIGURA 5.16 - Estereótipo e atributos que compõem as interfaces.

Há outros tipos de interfaces na *Web* como as interfaces de áudio e de vídeo. Porém estas interfaces são exibidas de forma independente, não fazendo parte da interface do *Web site*. Por isso não foram previstos estereótipos para estas situações. Os estereótipos mostrados podem também ser compostos de outros atributos, como por exemplo, na *home page* pode ter um formulário, na barra de navegação podem conter imagens. Os estereótipos representam os atributos mais freqüentes.

Na maioria das vezes não há uma resposta precisa para todas as questões que envolvem assuntos e interfaces e nem há questões suficientes para se obter todo tipo de respostas, uma vez que novos assuntos surgem a cada dia e a importância de um conteúdo pode ser diferente de tempos em tempos. O uso de questionários ou *chek list* na análise de requisitos serve como forma de orientação para a ordenação dos assuntos em busca de conteúdos mais coerentes.

5.2.3 - Definição do Tipo de Estrutura

Para que as interfaces do *Web site* tenham uma estrutura padrão, o PDW-UML propõe um modelo de Estrutura Dinâmica (ED) que envolve um servidor de páginas ativas; a organização de arquivos e diretórios; o uso de linguagens como HTML, JavaScript, CSS e uma linguagem dinâmica com recursos de SSI como PHP, por exemplo. Este modelo de

estrutura prevê os diretórios “abertos” (que são as portas de acesso ao usuário) criando um diretório especifico com arquivos-matriz. Os arquivos-matriz formam e padronizam a estrutura das interfaces, independente de como seja elaborada a estrutura lógica do sistema ou a estrutura (*design*) do conteúdo (textos, imagens e *links*) que compõe a interface.

Os arquivos que trazem as instruções de *design* de textos ou de conteúdo (*links*, textos e imagens) podem ter extensões html, txt, js, css etc. Entretanto os arquivos que têm instruções para chamar outro para que lhe seja parte integrante (instruções de diretivas de *include*) precisam ter extensão PHP, pois se trata de um recurso que precisa ser interpretado pelo servidor de páginas PHP.

A Figura 5.17 mostra a organização, o *script* e o *design* de uma interface desenvolvida com ED.

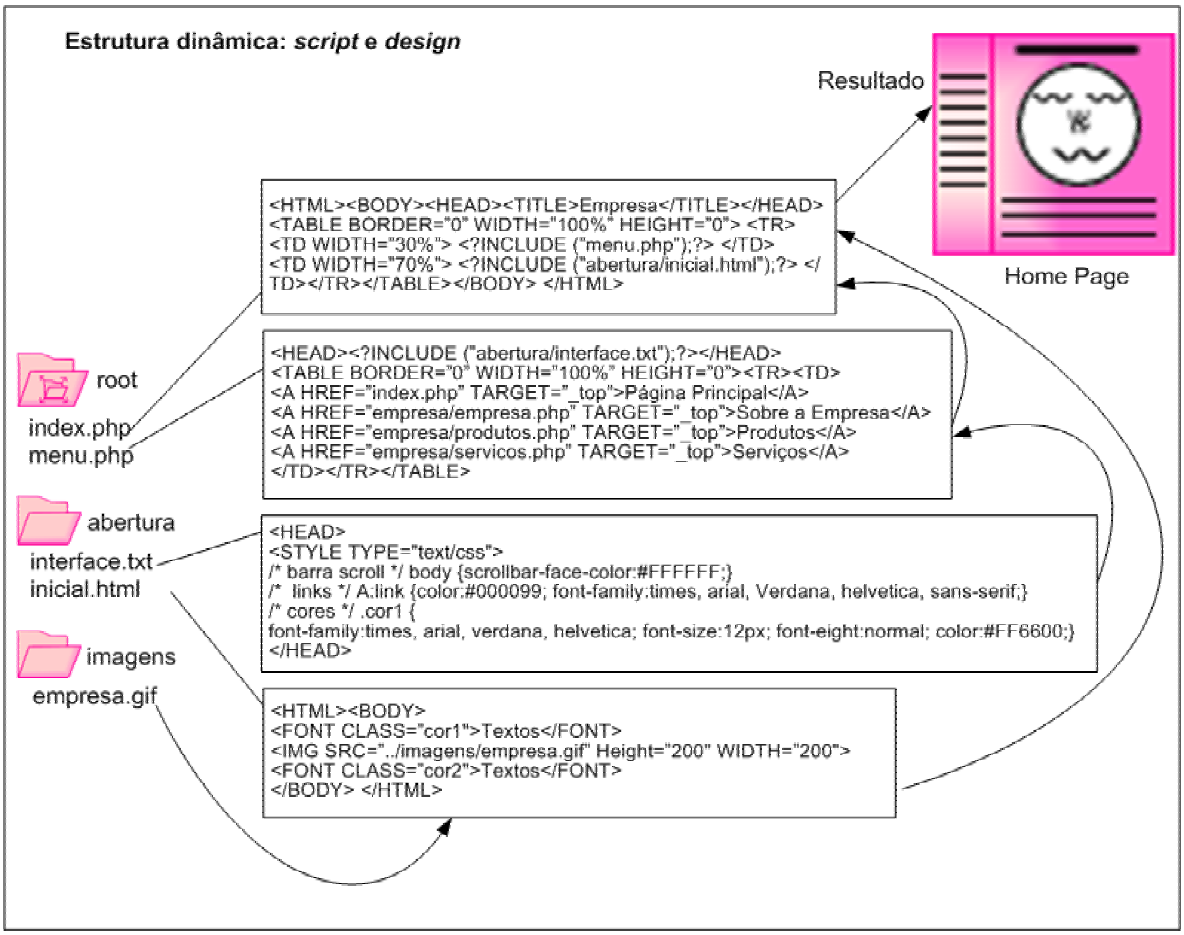


FIGURA 5.17 - Organização, *script* e *design* de uma interface com ED.

Este modelo de estrutura é composto por arquivos-matriz, que são arquivos onde não há a exposição direta de conteúdo, mas a chamada de arquivos com conteúdo que farão a

composição das interfaces. Dessa forma a barra de navegação (ou outro conteúdo necessário) fica sempre visível, porém sem a necessidade de que o *script* esteja diretamente em todas os arquivos e sim em um único arquivo que é chamado pelos arquivos-matriz.

O dinamismo no desenvolvimento pode ser considerado devido às possibilidades de reusabilidade (facilidade de reutilizar, usar novamente) o conteúdo. O conteúdo (textos, *links* e imagens) de cada interface fica incluso em arquivos-matriz onde não há código exposto diretamente, mas as divisões feitas com recursos de tabelas e as instruções de inclusão de arquivos. Em um único arquivo são feitas as instruções de *design* de textos (cores, fontes, tamanhos, alinhamentos etc.), este arquivo é incluso no arquivo-menu e o arquivo-menu é incluído em todos os arquivos-matriz. Com isso reduzem-se as instruções de inclusão e permite que os arquivos de conteúdo tenham o mínimo de instruções de *design* possível, de forma que possam ser reutilizados em outros Web sites e com outras tecnologias, apenas com pequenas modificações.

A seguir são mostrados alguns exemplos com os *scripts* que formam as EDs (divisão com tabelas e diretivas de *includs* arquivos de design e de conteúdo).

A Figura 5.18 mostra uma divisão horizontal formando duas células. Na célula esquerda está incluída a barra de navegação e na célula da direita o conteúdo da *home page*.

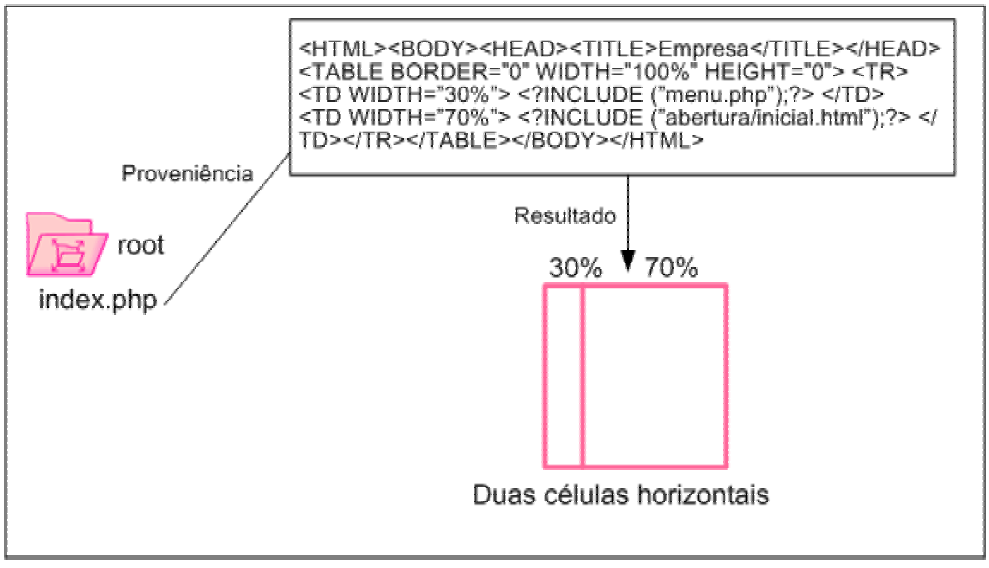


FIGURA 5.18 - Divisão horizontal com duas células.

O exemplo mais tradicional de páginas *Web* é uma barra de navegação vertical à esquerda, com aproximadamente 30% do tamanho da tela, sendo o restante destinado ao conteúdo propriamente dito. Esta estrutura, segundo estudos ergonômicos, segue o mesmo

procedimento de uso do livro e do caderno, o que faz com que o usuário não dispenda de muitos esforços para entender a interface como um todo.

Para este exemplo, com o uso de ED, são necessários três arquivos: o arquivo com a barra de navegação, o arquivo da página principal e o arquivo-matriz que faz a chamada de ambos, resultando em uma única página, a qual é chamada pelo usuário.

- arquivo 1: default.php
- arquivo 2: menu.php
- arquivo 3: abertura/inicial.html

Independente de quantos *links* haja na barra de navegação ou como seja o *design* e o conteúdo do arquivo que ocupa a célula a direita, já existe uma estrutura definida. Assim como foi feito para a interface inicial, é só utilizar o recurso de "Salvar Como", para duplicá-los e obter a mesma estrutura para as demais interfaces que façam parte do domínio. Quando os arquivos-matriz são duplicados, mantém-se a barra de navegação e troca somente o arquivo (e o diretório quando é o caso) para compor a célula da direita como, por exemplo, "abertura/inicial.html", passa a ser "funcionarios/funcionarios.html", todo o restante permanece igual, garantindo o padrão da estrutura.

A Figura 5.19 mostra uma segunda forma de estrutura formada por uma divisão horizontal formando três células. Neste exemplo considera-se que esta interface estará em um segundo nível e não mais no nível *root* como mostrado na Figura 5.18.

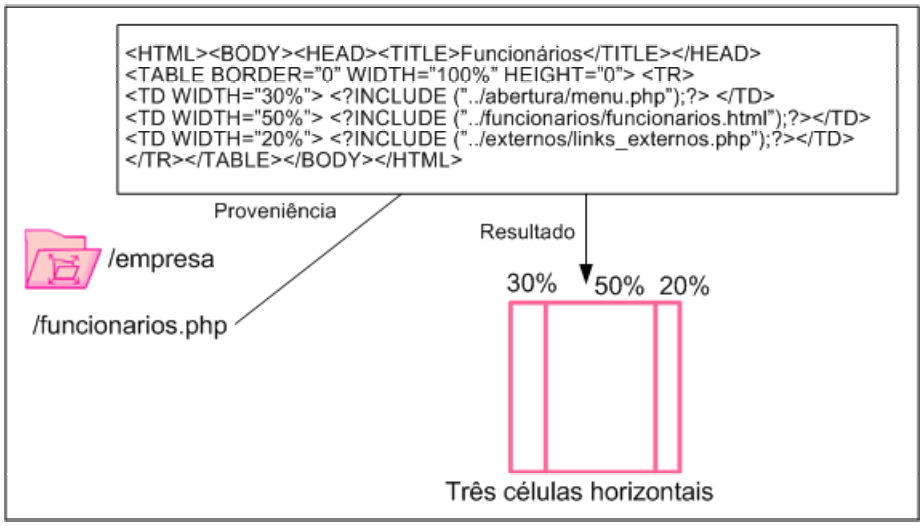


FIGURA 5.19 - Divisão horizontal com três células.

Como somente a interface inicial é mantida no diretório *root*, desenvolve-se o primeiro arquivo-matriz em um diretório de segundo nível, troca-se o endereço, o nome do diretório e

o nome dos arquivos. A partir daí basta novamente utilizar o recurso de "Salvar Como", nomear o arquivo e trocar o nome do diretório e do arquivo que irão compor as células.

Independente de qual seja a necessidade de divisões, uma vez que seja estabelecida, no nível *root*, basta copiar para um diretório de segundo nível (sejam todos os arquivos-matriz em um mesmo diretório no segundo nível ou na raiz de vários diretórios que também estejam no segundo nível) e trocar apenas o endereço e nome dos arquivos, mantendo o mesmo padrão de estrutura.

No exemplo da Figura 5.19 aparecem três arquivos que irão compor o conteúdo das três células existentes na estrutura do arquivo-matriz que está em um diretório chamado "empresa", no mesmo nível (segundo nível) dos diretórios "abertura, funcionarios, e externos":

- arquivo 1: ../abertura/menu.php
- arquivo 2: ../funcionarios/funcionarios.html
- arquivo 3: ../externos/links_externos.html

Como os quatro diretórios estão no mesmo nível basta utilizar a instrução "../" (sobe um nível) e o nome do diretório (e entra no diretório). Considerando que todos os arquivos-matriz estão no diretório "empresa", então o *script* diz para o sistema: "suba um nível, entre no diretório x e faça uso do arquivo mencionado".

O exemplo da Figura 5.19 é uma das maneiras mais simples de organização de diretórios, pois se houver necessidade de mudança na estrutura, basta entrar em um arquivo, fazer as modificações e fazer as repetições para os demais, sem a necessidade de abrir outros diretórios ou de ficar procurando diretórios que são acessados pelo usuário como na forma convencional de SSI. Porém o PDW-UML considera também algumas inconveniências em se manter um grande número de arquivos no mesmo diretório, ainda que todos estes arquivos sejam praticamente iguais, como é o caso dos arquivos-matriz.

Quando o *Web site* (mais precisamente os grandes portais) tem segmentos que sugerem nomes de arquivos iguais, mantendo todos os arquivos-matriz em um mesmo diretório anularia a convenção de utilizar nomes de arquivos mnemônicos. Considerando, por exemplo, que um domínio "empresa" tenha um segmento que se chama "produtos", e outro que se chama "serviços". Estes dois (ou mais) segmentos têm um sub-item que se chama "catalogo". Para manter a convenção de usar nomes mnemônicos seriam necessários dois arquivos-matriz com o mesmo nome. Para situações como esta o sistema de manter todos os

arquivos-matriz em um mesmo diretório se torna inconveniente, requerendo a segunda forma de organização prevista pelo PDW-UML, conforme mostrado na Seção 5.2.4.2.

Um outro inconveniente de manter todos os arquivo-matriz em apenas um diretório é que os URLs são formados pelo nome do diretório seguido do nome do arquivo. Quando os endereços de *Web site* são divulgados por outros meios de comunicação como, jornais, rádio, TV etc. onde o usuário precisa anotar ou memorizar o endereço. Nesse caso, quanto mais curtos forem os URLs, os usuários têm mais facilidade para memorizar, bem como para anotar e digitar posteriormente.

A Figura 5.20 mostra uma forma de divisão horizontal e vertical formando três células. Para esta estrutura considera-se o sistema de organização onde cada arquivo-matriz encontra-se na raiz do diretório específico de seu assunto, em um segundo nível.

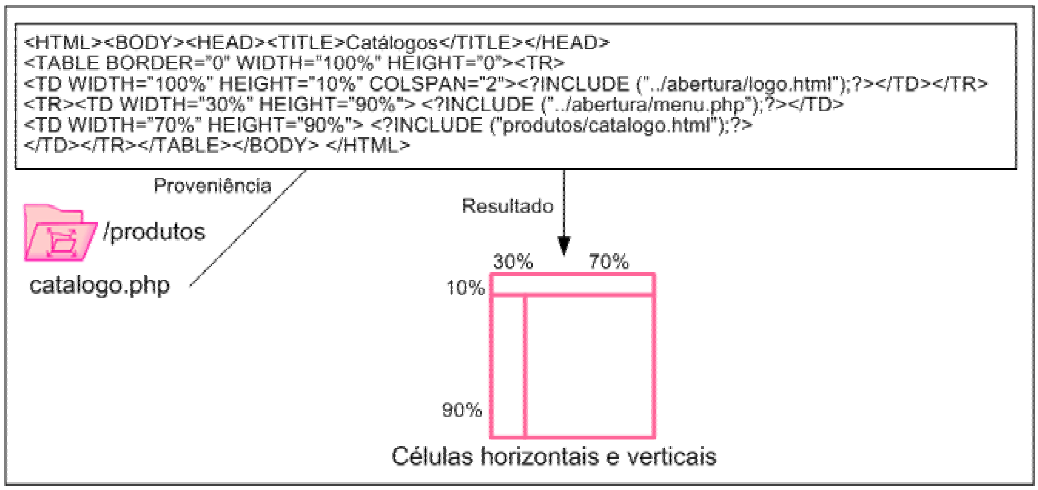


FIGURA 5.20 - Divisão horizontal e vertical formando três células.

No exemplo da Figura 5.20 o arquivo-matriz, chamado de “catalogo.php” está na raiz do diretório “produtos”, que se encontra em um segundo nível. Este arquivo tem uma estrutura formada por três células sendo uma horizontal e duas verticais. Estas células são compostas pelos seguintes arquivos:

- arquivo 1: ../abertura/logo.html
- arquivo 2: ../abertura/menu.php
- arquivo 3: produtos/catalogo.html

A diferença que há neste exemplo é que os arquivos-matriz não estão todos em um único diretório, mas na raiz do diretório específico de cada assunto. Dessa forma o “arquivo 1” e o ”arquivo 2”, seguem o mesmo exemplo do modelo anterior e o arquivo 3 tem seu endereço

alterado pois o arquivo-matriz o chama a partir de um sub-diretório existente no diretório "produtos".

Independente de qual seja a forma de organização, os subdiretórios com seus respectivos arquivos existem no sistema, cada um com seu respectivo assunto, a diferença encontra-se em fazer com que os arquivos-matriz (que são os arquivos que os usuários acessam) estejam em um diretório específico para os arquivos-matriz, ou na raiz do seu diretório específico logo abaixo do *root*.

5.2.4 - Definição dos Níveis de URLs

Além da ED o PDW-UML visa também a redução dos níveis de URLs e segue um princípio de usabilidade que visa uma página para cada interface e um endereço (URL) para cada página. A forma de organização que mantém todos os arquivos-matriz em um mesmo diretório facilita o trabalho de desenvolvimento e administração. No entanto, para o usuário, independente de qual seja a forma de organização os níveis de URLs serão sempre o *root* ou um nível abaixo.

Os URLs (*Uniform Resourse Locator*) são a nomenclatura utilizada na Internet para indicar o endereço de um documento. Essa nomenclatura inclui três componentes: o protocolo, o endereço do servidor e a localização do arquivo. Por exemplo, no URL http://www.empresa.com.br/default.php, o http representa o protocolo, o www.empresa.com.br representa o servidor que será acessado e o default.php é a localização do arquivo específico (Chase, 2000).

Um dos problemas encontrados nos URLs são os tamanhos gerados devido aos diversos níveis de diretórios criados para organizar o sistema lógico. Por exemplo, um URL proveniente de um terceiro nível a partir do *root*, ficaria com um tamanho de nomenclatura semelhante a este:
http://www.empresa.com.br/produto/catalogo/feminino/catalogo.php.

Se o arquivo estivesse em um nível abaixo, somar-se-ia ao endereço, mais um nome de diretório e assim sucessivamente (Baker, 2001).

É indiscutível a necessidade da criação de subdiretórios para se obter uma boa organização lógica. O que o PDW-UML propõem é que o usuário seja poupado da necessidade de ter que

trabalhar com URLs longos, fazendo com que os endereços a serem acessados sejam somente do nível *root* e de apenas um nível abaixo onde ficam os arquivos-matriz.

Para ilustrar a forma de organização de arquivos e diretórios prevista pelo PDW-UML e a partir de quais diretórios os arquivo são acessados são mostrados alguns estereótipos na Tabela 5.4.

TABELA 5.4 - Estereótipos dos diretórios.

Estereótipo do diretório	**Descrição**
Diretório aberto que contém arquivo-matriz.	Os diretórios que contêm os arquivos-matriz são os diretórios cujo nome aparece no *location* do navegador, pois são os diretórios que contêm os arquivos que são acessados pelos usuários. Assim, são chamados de diretórios abertos e trazem no estereótipo uma pasta aberta com o distintivo de uma outra pasta aberta com setas apontando para várias direções. Os diretórios abertos aparecem somente no primeiro e segundo nível, pois representam os tamanhos dos URLs.
Diretório fechado que contém sub-diretórios e arquivos que compõem os arquivos-matriz.	Os diretórios fechados que contêm sub-diretórios são representados por uma pasta aberta, indicando que há sub-níveis ou sub-diretórios. Os diretórios fechados podem ter quantos níveis forem necessários para facilitar a organização, pois não representam URLs.
Diretório fechado (de último nível) que contém arquivos, mas não contém subdiretórios.	Os diretórios fechados que não contêm sub-níveis ou sub-diretórios são representados através de uma pasta fechada indicando que não há sub-diretórios a partir dele.

5.2.4.1 - URLs a Partir de um Único Diretório

O acesso a partir do diretório *root* representa o acesso direto ao domínio como, por exemplo www.empresa.com.br. Quando se acessam arquivos que estejam em subdiretórios esses são seguidos de uma barra, o nome do diretório, outra barra e o nome do arquivo (para o acesso a um arquivo que esteja em um diretório de segundo nível). Considerando que todos os arquivos-matriz estão em um único diretório chamado “empresa”, independente de em quantos níveis abaixo estejam os conteúdos que formam a interface (arquivo-matriz) os

URLs terão o nome do domínio, uma barra, o nome do diretório com os arquivos-matriz, outra barra seguida do nome do arquivo-matriz.

A Figura 5.21 mostra a forma de organização onde os arquivos-matriz encontram-se no diretório *root* e em um sub-diretório específico chamado "empresa". Os URLs previstos encontram-se do lado direito da Figura.

FIGURA 5.21 - Arquivos-matriz em um único diretório.

Conforme mostrado na Figura 5.21, o usuário terá acesso somente a partir de dois diretórios, o *root* e o "empresa" os quais contêm os arquivos-matriz que formam as estruturas padrões criadas através do uso de tabelas para fazer as divisões e do uso de SSI para fazer a inclusão dos arquivos que estão nos demais diretórios.

Independente de onde estejam os arquivos com os conteúdos que compõem os arquivos-matriz, os URLs são representados pelo nome do domínio, o nome do diretório e o nome do arquivo, sucessivamente, sempre no primeiro ou segundo nível.

5.2.4.2 - URLs a Partir do Diretório Respectivo de cada Assunto

Conforme mostrado na Seção 5.2.4 quando há necessidade da reincidência de nomes para os arquivos-matriz, mantê-los todos em um único diretório impossibilita esta realização. Para manter o nível dos URLs reduzidos ao segundo nível, o PDW-UML prevê uma segunda forma de organização onde os arquivos-matriz ficam na raiz de seus subdiretórios, juntamente com os arquivos de conteúdo e demais subdiretórios.

A Figura 5.22 mostra a forma de organização onde os arquivos-matriz encontram-se na raiz de seus respectivos diretórios, em um segundo nível a partir do diretório *root*.

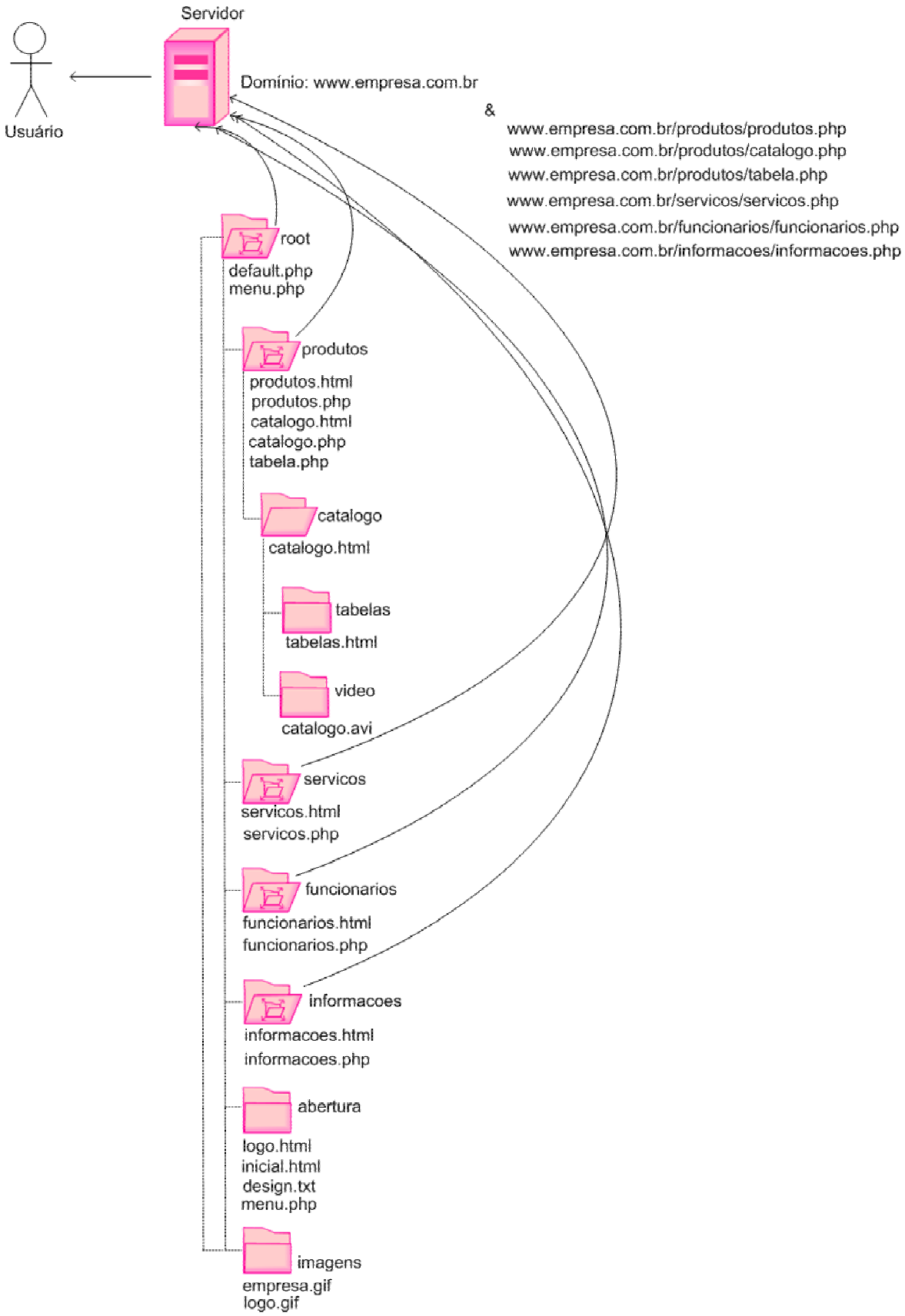


FIGURA 5.22 - Arquivos-matriz em diversos diretórios.

Além do exemplo mostrado na Figura 5.22 é possível reduzir ainda mais o tamanho dos URLs para **um dos** arquivos-matriz de cada diretório. Os arquivos com nome "*index*" ou "*default*", seguido de sua respectiva extensão (para este exemplo, default.php) são os arquivos que o servidor *Web* procura automaticamente quando um usuário digita o nome de um domínio ou o nome de um domínio seguido de uma barra e o nome do diretório. Isso significa que o servidor procura na raiz de cada diretório um arquivo com nome padrão como *index* e *default*, por isso quando estes estão na raiz de um diretório (independente do nível) não precisam ser mencionados.

Como na maioria das situações há mais de um arquivo-matriz pertencente a um mesmo diretório, esse recurso representa mais uma alternativa para divulgação de nomes de URLs ainda mais curtos. Para divulgação dos URLs onde há mais de uma interface em um mesmo diretório, ainda é necessário a informação do URL completo.

A Figura 5.23 mostra a forma de organização onde os arquivos-matriz encontram-se na raiz de seus respectivos diretórios, porém um dos arquivos-matriz tem seu nome (mnemônico) substituído pelo nome *default*, seguido de um ponto e sua extensão (default.php).

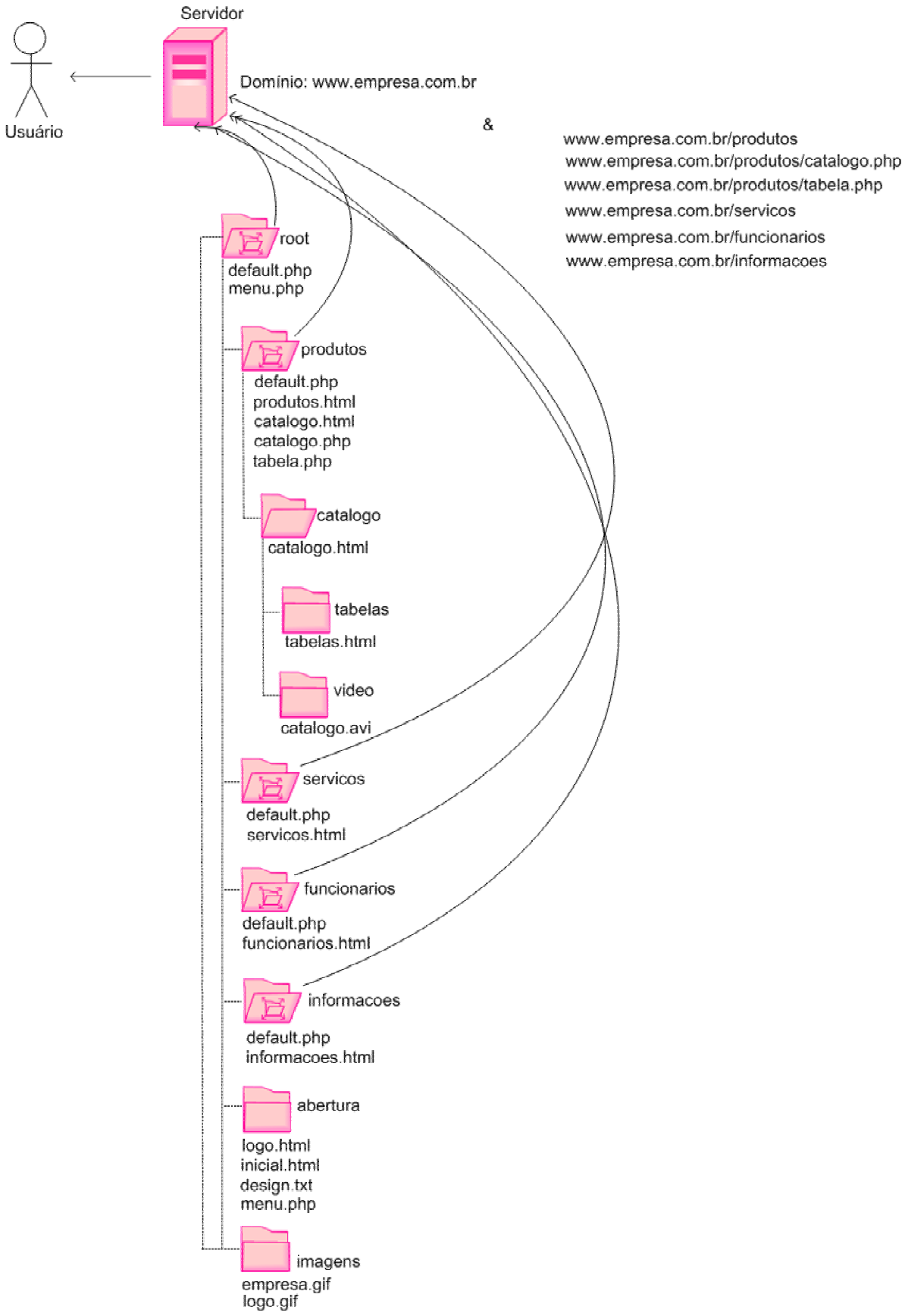


FIGURA 5.23 - Arquivos-matriz e arquivos-default em diversos diretórios.

Para os três exemplos de organização previstos a estrutura é a mesma, os níveis de URLs são os mesmos (primeiro e segundo). A escolha feita por um administrador ocorre a partir da identificação das necessidades de cada projeto como, por exemplo, o volume de arquivos, os nomes que podem ser mnemônicos para cada link e sua respectiva interface e meios de comunicação utilizados para a divulgação dos URLs.

O primeiro exemplo é o que simplifica mais o trabalho do administrador, pois no caso de mudança na estrutura, não há necessidade de percorrer vários diretórios.

5.2.5 - Composição das Interfaces

O PDW-UML considera algumas inclusões básicas para a composição das interfaces. Não se pode afirmar que todos os *Web sites* terão as mesmas necessidades, por isso o PDW-UML descreve interfaces estáticas e dinâmicas e de acordo com esta classificação mostra os *designs* mais freqüentes que é uma barra de navegação e um conteúdo referente a um *link*. Existem situações em que o conteúdo se divide em várias interfaces havendo a necessidade de uma sub-barra de navegação, nesse caso pode-se utilizar o mesmo exemplo mostrado para a barra de navegação das interfaces de primeiro e segundo nível, porém fazendo uma inclusão do sub-menu no arquivo de conteúdo. O restante segue o padrão das interfaces do modelo.

A Figura 5.24 mostra as bases de inclusão para composição dos arquivos-matriz, onde há um arquivo-matriz que traz em seu cabeçalho as *meta tags* do sistema, as divisões e suas respectivas instruções de *include*, formando as estruturas propriamente ditas.

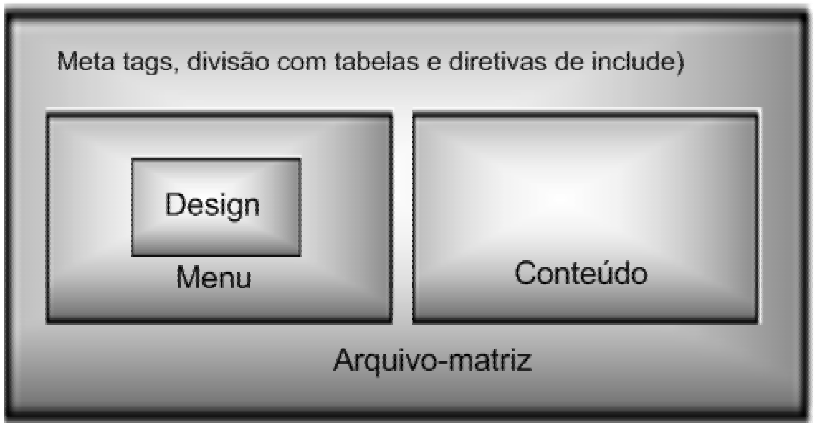


FIGURA 5.24 - Base de inclusão para composição dos arquivos-matriz.

A lógica existente na Figura 5.24 é que:

se o arquivo com o menu contém o arquivo de *design*;

e os arquivos “Conteúdo” e “Menu” estão inclusos no arquivo-matriz;

então os arquivos “Menu” e “Conteúdo” assumem as instruções de *design* existentes no arquivo “*design*”.

O PDW-UML utiliza CSS para caracterizar o *design* das interfaces (principalmente o *design* de textos), porém considera que nem todos os navegadores reconhecem todas as instruções de CSS. Quando há em CSS uma instrução que não é reconhecida por todos as navegadores, o PDW-UML substitui a instrução CSS por uma instrução HTML como, por exemplo, cores de fundo. Essa instrução é inserida diretamente na *tag* <BODY> dos arquivos “Menus” que se encontram repetidos no primeiro e segundo nível. Isso faz com que em caso de modificações o dinamismo e a redução de trabalho são mantidos, pois as modificações são feitas em apenas dois arquivos.

5.2.5.1 - Composição dos Arquivos-matriz

Os arquivos-matriz são compostos basicamente por *meta tags*, divisões feitas com tabelas e diretivas de *include*, conforme mostra a Figura 5.25 (arquivo default.php).

```
<HTML><BODY>
<HEAD>
<META http-equiv="Keywords" content="estrutura dinâmica para interfaces de web sites">
<META name="nome do domínio" content="conteúdo da página">
<META content="text/html; charset=iso-8859-1" http-equiv=Content-Type>
<!--fim instruções meta tags-->
</HEAD>

<TABLE BORDER=”0” WIDTH=”100%” HEIGHT=”0”> <TR>
<TD WIDTH=”30%”> <?INCLUDE (”menu.php”);?> </TD>
<TD WIDTH=”70%”> <?INCLUDE (”abertura/inicial.html”);?> </TD></TR> </TABLE>
</BODY> </HTML>
```

FIGURA 5.25 - Arquivo-matriz com *meta tags* no cabeçalho.

- ***Meta tags***: ao determinar os arquivos que serão acessados pelo usuário (arquivos-matriz), já se sabe automaticamente onde inserir as *meta tags*, com instruções de acordo com o conteúdo de cada página.

  Sistemas de buscas de páginas *Web*, como o *Google*, por exemplo, fazem uma varredura procurando em *meta tags* as palavras-chave digitadas pelo usuário. Com as *meta tags* inseridas nos arquivos-matriz, os arquivos que contêm as palavras

buscadas serão trazidas junto com a barra de navegação e as informações gerais do *Web site*, pois os arquivos-matriz, são compostos de *design*, barra de navegação e o conteúdo específico de cada interface.

- **Divisões**: independente de como e quantas sejam as divisões existentes nas interfaces com o conteúdo, a base da estrutura do *Web site* segue o mesmo padrão, pois todos os arquivos-matriz têm as mesmas divisões.
- **Diretivas de *include***: as diretivas de *include* juntamente com as divisões feitas com tabelas é que formam a estrutura do *Web site*, dentro das quais existem os conteúdos.

5.2.5.2 - Composição dos Arquivos de *Design*

Os arquivos de *design* são compostos por instruções de CSS que determinam quais são os estilos, tamanhos e cores de textos; características de tabelas; o ícone que aparece no *location* do navegador; as cores que caracterizam a barra de *scroll*; a cor de fundo do *body* e das tabelas etc.

No entanto, CSS é uma linguagem *client side*, que depende dos recursos do navegador do cliente para que suas instruções sejam executadas. Alguns recursos como, por exemplo, cor de fundo e características de tabelas não são reconhecidas por todos os navegadores. Assim, o PDW-UML utiliza HTML para instruções que se desenvolvidas com CSS podem comprometer o *design* e o entendimento do *Web site*. Um exemplo disso é o uso de cor de fundo das páginas que é feito com HTML e é inserido nos arquivos-menus. Como os arquivos-menus têm duas incidências, em caso de modificações é necessário que se alterem os dois arquivos (não apenas um como se estivesse no arquivo de *design* feito com CSS e nem em todos como se fosse feito sem o uso da inclusão do arquivo-menu nos arquivos-matriz).

A Figura 5.26 mostra um *script* com as instruções de *design* em CSS, como características de ícone, barras de scroll, efeitos em *links* para os estados “normal, sobre, pressionado e visitado” e instruções de fonte para tamanho, cor e estilo.

```
/* muda o ícone */
<LINK REL="shortcut icon" HREF="http://www.empresa.com.br/imagens/imagem.ico">

<STYLE type="text/css">
/*inicio body e scroll */
BODY {
scrollbar-3d-light-color:#000000;
scrollbar-arrow-color:#FFFFFF;
scrollbar-base-color:#FF6699;
scrollbar-dark-shadow-color:#000000;
scrollbar-face-color:#FF6699;
scrollbar-highlight-color:#FFFFFF;
scrollbar-shadow-color:#FF6699;
}
/*fim barra scroll */

/* inicio links */
A:hover {color:#000000; font-family:times, arial, Verdana, helvetica, sans-serif; font-size:14px;
background-color:#EAB4C4;}
A:link {color:#000000; font-family:times, arial, Verdana, helvetica, sans-serif; font-size:14px;}
A:visited {color:#000000; font-family:times, arial, Verdana, helvetica, sans-serif; font-size:14px;}
A:active {color:#FFFFFF; font-family:times, arial, Verdana, helvetica, sans-serif; font-size:14px;}
/* fim links */

/* inicio cor preta, exemplo normal e negrito, tamanho 14*/
.black14 {
font-family:times, arial, Verdana, helvetica, sans-serif;
font-size:14px;
font-weight:normal;
color:#000000;
}

.black14b {
font-family:times, arial, Verdana, helvetica, sans-serif;
font-size:14px;
font-weight:bold;
color:#000000;
}
</STYLE>
```

FIGURA 5.26 - *Script* com instruções de *design* em CSS.

5.2.5.3 - Composição dos Arquivos-menu

O PDW-UML considera a necessidade de duas incidências de um arquivo-menu ou arquivo com a barra de navegação: um no nível *root* e outro em um segundo nível. A seguir são mostrados exemplos de como o sistema interpreta os endereços lógicos em ambos os níveis.

- **No nível *root***: na Figura 5.27, na linha 2, há uma instrução de *include* que instrui o sistema para sair do diretório atual, entrar no diretório "abertura" e executar o arquivo "interface.txt". A mesma instrução vale para a inclusão das imagens nos *links*. Na linha 7 a instrução é para que seja executado o arquivo "default.php". Na linha 11 há instrução para sair do diretório atual, entrar no diretório "empresa" e executar o arquivo-matriz "produtos.php". Esta mesma instrução vale para os demais *links*, uma vez que os demais arquivos-matriz estão no diretório "empresa". Para o caso de manter cada arquivo-matriz em seu respectivo diretório, a instrução seria "../diretorio/arquivo-matriz.php", ou seja, sair do diretório atual, entrar em um outro diretório no mesmo nível e executar o arquivo-matriz.php.

  A Figura 5.27 mostra os endereços lógicos (internos) de um arquivo-menu que fica do diretório *root*.

1	<HEAD>
2	<? INCLUDE="abertura/interface.txt";?>
3	</HEAD>
4	<BODY BGCOLOR="#FFFFFF">
5	<TABLE BORDER="0" WIDTH="100%" HEIGHT="0%" CELLSPACING="0" CELLPADDING="0"><TR><TD NOWRAP>
6	<TR><TD NOWRAP>
7	<A HREF="default.php" TARGET="_top"><IMG SRC="imagens/imagem1.gif" BORDER="0" WIDTH="20" HEIGHT="17" ALIGN="top">Página Inicial</A>
9	</TD></TR>
10	<TR><TD NOWRAP>
11	<A HREF="empresa/produtos.php" TARGET="_top"><IMG SRC="imagens/imagem1.gif" BORDER="0" WIDTH="20" HEIGHT="17" ALIGN="top">Produtos</A>
12	</TD></TR><TABLE></BODY>

FIGURA 5.27 - Arquivo-menu no nível *root*.

- **No segundo nível**: na Figura 5.27 pode-se observar logo na segunda linha um endereço lógico com a instrução "../abertura/interface.txt" que significa dizer para o sistema: saia do diretório atual, suba um nível, entre no diretório "abertura" e execute o arquivo "interface.txt". A mesma instrução vale para as imagens que compõem os *links*. Na linha 7 é para sair do diretório atual, subir um nível e

executar o arquivo “default.php”. Na linha 11 a instrução é para que se execute o arquivo “produtos.php”. Essa mesma instrução será para os demais *links* uma vez que o arquivo-menu está no mesmo diretório que os arquivos-matriz de segundo nível. Se fosse utilizada a opção de manter os arquivos-matriz na raiz de seus respectivos diretórios o arquivo-menu poderia ficar no diretório que tem os arquivos de abertura, por exemplo. Nesse caso a instrução seria “../diretorio/arquivo-matriz.php”.

A Figura 5.28 mostra um arquivo-menu em um sub diretório de segundo nível.

1	<HEAD>
2	<? INCLUDE="../abertura/interface.txt";?>
3	</HEAD>
4	<BODY BGCOLOR="#FFFFFF">
5	<TABLE BORDER="0" WIDTH="100%" HEIGHT="0%" CELLSPACING="0" CELLPADDING="0"><TR><TD NOWRAP>
6	<TR><TD NOWRAP>
7	<A HREF="../default.php" TARGET="_top"><IMG SRC="../imagens/imagem1.gif" BORDER="0" WIDTH="20" HEIGHT="17" ALIGN="top">Página Inicial</A>
9	</TD></TR>
10	<TR><TD NOWRAP>
11	<A HREF="produtos.php" TARGET="_top"><IMG SRC="../imagens/imagem1.gif" BORDER="0" WIDTH="20" HEIGHT="17" ALIGN="top">Produtos</A>
12	</TD></TR><TABLE></BODY>

FIGURA 5.28 - Arquivo-menu em um subdiretório.

Conforme mostrado nesta Seção, os endereços dos *links* são feitos conforme a localização dos arquivos-matriz. Como o endereço do nível *root* para os subdiretórios é diferente do endereço dos sub diretórios para o nível *root* (diretorio/arquivo é diferente de ../diretorio/arquivo), a solução encontrada pelo PDW-UML foi manter um arquivo-menu no nível *root* e outro em um subdiretório, logo abaixo do *root*.

5.2.5.4 - Composição dos Arquivos de Conteúdo

O conteúdo de cada interface normalmente é composto de textos, imagens e *links*; somente textos, somente imagens etc. podendo ser feito em *softwares* como o Macromedia Flash,

Dreamweaver, Front Page, Editor VI, ou outro editor da preferência do desenvolvedor. O que precisa ser feito conforme instruções do PDW-UML são os endereços dos *links*.

- **Arquivos-matriz em um único diretório**: (conforme mostrado na figura 5.23) quando se está em uma interface em um terceiro nível (independente do endereço deste arquivo), para voltar para o nível anterior basta apontar o endereço do arquivo-matriz que representa a interface anterior como, por exemplo, <A HREF=”anterior.php” TARGET=”top”>Página Anterior</A>. O mesmo exemplo vale para quando se deseja ir de um segundo nível para um terceiro, quarto, quinto etc. O sistema entende que um arquivo-matriz chama outro arquivo-matriz, por isso os *links* são sempre para o arquivo-matriz que tem a interface com o conteúdo em questão. Portanto no arquivo-matriz se faz o *include* do arquivo de conteúdo e no arquivo de conteúdo se faz chamada ao arquivo-matriz, ou seja, os endereços de *links* são de uma interface para outra interface e o PDW-UML considera que cada interface é representada por um arquivo-matriz.

  Quando, de um arquivo de conteúdo (independente de qual seja o nível onde este se encontre), se deseja fazer um *link* para o arquivo “default.php” que é o arquivo-matriz que está no diretório *root*, basta instruções para sair do diretório do arquivo-matriz que representa o conteúdo em questão e apontar um nível acima como, por exemplo, <A HREF=”../default.php” TARGET=”top”>Página Inicial</A>.

- **Arquivos-matriz em vários diretórios**: (conforme mostrado na figura 5.24) quando os arquivos-matriz ficam na raiz de seus respectivos diretórios, para voltar ao nível *root* utiliza-se o mesmo endereço que se usa quando os arquivos-matriz estão em apenas um diretório como, por exemplo, <A HREF=”../default.php” TARGET=”top”>Página Inicial</A>. O que muda é que ao invés de o endereço dos *links* serem de um arquivo-matriz para outro arquivo-matriz o endereço é apontado para um nível acima e em seguida para a entrada no respectivo diretório como, por exemplo, <A HREF=”../diretorio/anterior.php” TARGET=”top”>Página Anterior</A>.

  Para ambos os tipos de organização, quando se têm arquivos para *download* ou arquivos com a extensão pdf, por exemplo, deve ser utilizado o endereço real onde se encontra o arquivo, considerando a saída a partir do endereço do arquivo-matriz e

o parâmetro TARGET recebe valor = _blank. Mesmo que o arquivo para *download* ou pdf esteja no mesmo diretório que o arquivo de conteúdo, o sistema entende o endereço do arquivo-matriz, o que significa que o endereço não será apenas a referência ao arquivo que está no mesmo diretório, mas a partir do arquivo-matriz para o endereço do arquivo para download como, por exemplo, <A HREF="../downloads/arquivo.zip" TARGET="_blank">Arquivodownload.zip</A>.Ou <A HREF="../downloads/arquivo .pdf" TARGET="_blank">Arquivo .pdf</A>

Para evitar URLs longas o PDW-UML considera a existência de um diretório, no segundo nível, específico para arquivos .pdf e arquivos para *download*.

- **Interfaces contextuais**: quando se trabalha com interfaces dinâmicas resultantes de consultas a bancos de dados, uma consulta gera uma interface e partir dessa interface outras são geradas, conforme necessidade de uma operação. Para essas situações o sistema de *link* é feito de uma interface para suas interfaces sucessoras, com o endereço de um arquivo-matriz para outro. Assim como, para as interfaces permanentes, elabora-se um arquivo-matriz e um arquivo com um conteúdo específico que no caso das interfaces contextuais, normalmente traz instruções permanentes e instruções para recepção do resultado da consulta realizada. Uma seqüência de interfaces ocorre a partir de um *link* que corresponda a uma interface com um formulário, como ocorre em interfaces de inserção de dados; interface que armazena dados temporários; interface de envio de dados para o banco de dados e; interface informativa do status da operação.

  - **Interface de inserção de dados**: normalmente as interfaces contextuais são geradas a partir de uma interface que traz um formulário a partir do qual o usuário consulta, envia ou recebe informações. Para fazer um *link* para a interface seguinte informa-se o endereço do arquivo-matriz correspondente à interface seguinte como, por exemplo, <FORM ACTION="arquivo-matriz1.php" METHOD="post"> (na maioria dos casos o método *post* é utilizado para envio de dados e o método *get* para consultas). O resultado da interação será mostrado na interface seguinte, juntamente com o *design* e a barra de navegação que estão inseridos em todos os arquivos-matriz.

- Interface que armazena dados temporários: os dados temporários podem ser itens que são depositados em carrinhos de compra ou dados que são armazenados para simples correção antes que sejam gravados em um banco de dados. Se os arquivos-matriz estiverem em endereços diferentes, menciona-se o endereço e se estiverem no mesmo diretório basta novamente que o valor do parâmetro "action" seja o nome do arquivo-matriz como, por exemplo, <FORM ACTION="arquivo-matriz2.php" METHOD="post">.

- Interface de envio de dados para o banco de dados: uma interface com instruções para envio de dados pode ser a interface onde aparece um primeiro formulário; a segunda após o formulário; a primeira, segunda, terceira etc. após uma interface que armazena dados temporários. O *link* se faz da mesma forma que para as interfaces anteriores apontando o arquivo-matriz da interface seqüente para o parâmetro "action" do formulário: <FORM ACTION="arquivo-matriz3.php" METHOD="post">.

- Interface informativa do status da operação: normalmente após a conclusão de uma operação ou após o envio de dados mostra-se uma interface trazendo informações do resultado da operação realizada. No caso de pesquisa e consultas em geral, esta interface aparece logo após a página com o formulário inicial com o resultado da consulta. Quando representa o envio de dados a um banco de dados mostra-se a operação foi concluída com sucesso ou não. Além da mensagem informativa, normalmente essa interface traz um ou vários *links* como opções para que o usuário possa escolher suas próximas operações.

5.2.6 - Testes do Sistema Navegacional

Mesmo que cada *link* tenha sido testado logo após sua implementação na máquina cliente, antes de iniciar a fase de *design* das interfaces considera-se a necessidade de um teste navegacional do sistema como um todo, a partir da máquina servidora. A engenharia *Web* aponta os erros de formulação dos *links* como os maiores problemas de navegação dos sistemas *Web*.

Como o PDW-UML considera a existência da incidência da barra de navegação em dois níveis do sistema lógico, então os testes precisam ser feitos em todos os *links* a partir do

diretório *root* e em todos os *links* a partir de uma interface qualquer que represente o segundo nível. Os demais testes precisam ser feitos a partir do acesso a todas as interfaces que tenham *links*, quer para URLs internos ou externos.

Quando há interfaces dinâmicas no sistema fazem-se os testes a partir de consultas e envio de dados, testes de uso da documentação *on-line* e estudos de tratamentos de erros, se detectados.

5.3 - Fase 3: D*esign* das Interfaces

As duas fases anteriores são feitas através dos recursos da UML, da Engenharia de *Software* (ES) e da Engenharia *Web* (EW). Nesta fase inicia-se um processo onde as soluções não são encontradas na UML ou na engenharia, porque cada *Web site* requer uma solução diferente. As duas fases anteriores já representam a parte funcional do *Web site*, com o uso da UML e da engenharia para fazer modelagem e a implementação. A engenharia já cumpriu seu papel ao fornecer estudos formalizados para levantar, organizar e selecionar requisitos; para modelar o sistema; para fazer testes funcionais sob diversas perspectivas; para um desenvolvimento do sistema que permita usabilidade para o usuário e reusabilidade para a equipe; para organizar os arquivos e diretórios de forma que os URLs sejam reduzidos sem comprometer a organização dos assuntos que pertencem ao domínio na máquina local. Enfim, o uso de recursos formalizados para o desenvolvimento do projeto, para a implementação do *Web site* e para os testes realizados, pode ser mostrado como um trabalho baseado na engenharia.

A fase de *design* das interfaces é proposta como uma fase artística que busca aprimorar o que foi desenvolvido nas fases anteriores, propondo um tempo e um espaço dentro de um processo para que haja a realização artística e criativa. A proposta para a fase de *design* é uma forma de cultura que envolve estética, arte e ética como base para um estilo de *design*.

A arte na *Web* pode ser vista e entendida como o domínio que tem um *Web designer* com as ferramentas gráficas utilizadas para a *Web* e no senso (cultura) ético e estético que utiliza para compor um *design* final. Assim, a diferença entre a arte e a engenharia utilizadas em *Web sites* pode ser representada pela finalidade e pela formalização. A finalidade da engenharia é projetar e implementar sistemas funcionais, formalizados e padronizados. A arte

tem como finalidade tratar, personalizar e melhorar a estética gerada pela engenharia de forma que cada *Web site* possa ter uma estética única, de acordo com o objetivo da cada domínio.

Assim, não se propõe um padrão de *design,* pois poderia representar uma falta de reconhecimento da capacidade de criação dos profissionais da *Web* e da necessidade mercadológica de apresentar os diferenciais competitivos que um *design* diferenciado pode oferecer. Ainda há a questão que é puramente de "gosto", maturidade visual etc. que leva a questionar qual seria o perfil dos *Web designers* que fariam uso de um padrão de *design* proposto, seja em um mprocesso de desenvolvimento, *template*, protótipo etc.

O *design* não atrapalha o bom funcionamento. O que pode atrapalhar é a forma inadequada do uso das tecnologias como, por exemplo, o uso inadequado de um *software* de compactação de imagens pode deixá-las pesada demais e retardar o tempo de carregamento. Não é a arte e, muito menos a criatividade (utilizada para que uma imagem expresse um objetivo, nas combinações de cores, tamanhos e alinhamentos) que interferem no bom funcionamento.

O usuário entende a arte e o *design*. O usuário normalmente não entende e não tem que entender a engenharia usada para o desenvolvimento. Então o que se propõe nesta fase é que se disponha ao usuário interfaces com *designs* artísticos (quando necessário), criativos, mas orientados ao objetivo (assunto) da interface e ao domínio como um todo. E, isso não atrapalha o funcionamento navegacional e nem retarda o tempo de carregamento.

Algumas questões são apresentadas como base de uma cultura que pode ajudar um *Web designer* a planejar o *design* de interfaces e a fazer escolhas em caso de dúvidas entre um *design* e outro; entre uma estrutura e outra; entre uma tecnologia e outra etc. Para elaborar a interface humana, além do conhecimento da engenharia, consideram-se também os benefícios que o conhecimento de algumas ciências humanas podem proporcionar.

A psicologia e a filosofia têm colaborado com algumas ciências exatas como, por exemplo, com a robótica e com a cibernética, para auxiliar no desenvolvimento de sistemas que facilitam as atividades humanas através da inteligência artificial (Roldão, 2000). A partir dessas observações é que se propõe, para o desenvolvimento de *design* de interfaces *Web*, o conhecimento de alguns aspectos da psicologia, mais precisamente da psicologia cognitiva e

da filosofia, mais precisamente da arte, da ética e da estética. Questões mercadológicas não serão abordadas neste trabalho.

Esta fase requer uma análise do conteúdo que está disponível ao usuário para buscar uma harmonia entre os atributos; é onde se analisa como ficará a distribuição dos *links* na *home page* e nas demais interfaces; o que deve e o que não deve aparecer na *home page*; o espaço que cada imagem ocupará em uma interface, independente de seu peso; as frases e palavras atribuídas aos *links*, independente dos nomes dos arquivos; em que parte de um texto deve-se inserir um *link* para um outro assunto ou complemento do assunto em questão; qual o peso máximo para cada imagem uma vez que tenha sido definido seu tamanho, (largura x altura); o espaço reservado aos atributos que ficam fixos na interface e ao conteúdo que se renova a cada *link*; o *design* dos textos; os alinhamentos dos textos em relação às imagens e em relação aos títulos e sub-títulos.

Desenvolver *Web sites* como "processos artísticos" é diferente de desenvolver com arte e criatividade interfaces que tenham sido desenvolvidas como um processo de engenharia e que no seu devido tempo e espaço receberam um tratamento diferenciado. O conceito de arte para a *Web* vai além da arte utópica e engloba a arte objetiva, que representa o uso da educação artística expressa através do uso de ferramentas para a *Web* e do senso ético e estético.

A abordagem sobre a ética não visa fazer julgamentos ou buscar a culpas ou intencionalidade de bons e maus *designers*, mas é apontada como uma sugestão de estudo ou conhecimento de um assunto que pode ajudar em uma tomada de decisão. Da mesma forma conhecimentos sobre a estética são abordados como um posicionamento do usuário diante de uma estética; das impressões causadas por uma estética, principalmente quando uma estética é revelada no conceito de belo, conforme o que o belo pode representar a cada usuário.

A engenharia, mais precisamente a Engenharia *Web*, cumpre seu papel em um processo de desenvolvimento ao oferecer respaldo a engenheiros e demais profissionais da *Web*. O papel da engenharia é oferecer recursos e soluções baseados em experimentos que tenham sido aprovados e mostrados através de formalidades que podem ser conceituais, práticas, descritivas e/ou ilustrativas.

Um sistema *Web* pode ser desenvolvido apenas com base em princípios da engenharia, mas isso tornaria a *Web* menos atraente a alguns perfis de usuários e a alguns segmentos mercadológicos. Conforme ocorre na área das ciências humanas, biológicas etc., na área das

exatas, as ciências também podem se complementar para proporcionar aos usuários dos produtos e serviços em questão, uma qualidade maior.

Normalmente as pessoas que estão envolvidas em um projeto têm um certo domínio de suas atividades e sabem o que, para que e para quem está sendo feito, mas em sistemas *Web* os usuários são os mais diversos possíveis, com os mais diversos níveis de conhecimentos e habilidades para com o uso de computadores. O PDW-UML considera que não cabe somente a engenharia se encarregar de levar ao usuário a melhor forma de entendimento, atrativos mercadológicos, *designs* atraentes e demais questões que possam atrair a atenção do usuário e lhe proporcionar o lazer e as informações procuradas.

Existem várias maneiras de expressar uma mesma idéia, de narrar um mesmo fato, bem como de descrever um produto ou um serviço. Há escritores que utilizam páginas e mais páginas para expressar uma idéia básica (isso se vê principalmente em obras literárias), mas na *Web* essa forma de expressão pode não ser a mais adequada devido ao tempo de carregamento de uma página; da paciência do leitor em ficar lendo na tela, clicando em *links* ou usando a barra de rolagem; das técnicas de redação para a *Web*, que são diferentes das técnicas utilizadas para recursos onde o volume de páginas não representa problemas.

O PDW-UML, na fase de *design* das interfaces, considera combinações de expressões em textos e imagens, cores, alinhamentos (sons e vídeos, se necessários) como uma saída para expressões que se expostas sempre na forma escrita ou na forma ilustrativa poderiam parecer incompletas; se expostas sempre na forma original, conforme o padrão do navegador, não ofereceriam espaço para personalizações. A maneira como estas combinações de textos, imagens, *links* e formulários podem ser expostos é o trabalho que se realiza nesta terceira fase.

A seguir são mostrados alguns conceitos filosóficos e tecnológicos utilizados como base para a proposta da terceira fase.

5.3.1 - Arte e Técnica como Princípios de *Design*

O *design* representa literalmente o desenho que é apresentado em uma interface. Este "desenho" é desenvolvido através do uso de técnicas em ferramentas de desenvolvimento e tratamento de imagens e técnicas de uso de ferramentas de design para interfaces Web. No entanto, com o uso das mesmas técnicas, o resultado poderá ser diferente dependendo da

maturidade e da educação artística de cada profissional. O resultado obtido através de técnicas de uso de ferramentas representa a arte utilizada na Web (Domingues, 2003).

O desenvolvimento de *Web sites* requer conhecimentos sólidos de técnicas de desenvolvimento, bem como de estratégias para obter melhores resultados das tecnologias utilizadas. No entanto, se na fase de *design* das interfaces, todos os profissionais da *Web* seguissem os mesmos critérios e tivessem a mesma visão, a tendência seria anular ou deixar de desenvolver a inovação, a criatividade e a competitividade entre profissionais ao oferecer serviços diferenciados.

Dentro da filosofia, há situações em que a ética, a estética e a arte são abordadas de forma muito semelhante, pois ambas estudam formas de entender e desenvolver atividades que proporcionem formas de compreensão e satisfação a expectadores e usuários (Pauli, 1997). A ética representa uma busca constante de se fazer melhor aquilo que se faz; a estética como uma forma de afirmação e concretização de pensamentos e percepções subjetivas e a arte como uma educação artística abrangente que engloba a ética e a estética.

Para a fase de *design* das interfaces essas abordagens são feitas com o objetivo de "buscar" levar até o usuário o melhor *design* possível dentro dos limites de um projeto, independente se os resultados sejam provenientes de técnicas, da arte, da estética ou da ética ou de uma maturidade proveniente destas e de outras fontes. A seguir são mostrados maiores detalhes sobre a arte, a estética, a ética e algumas formas de uso na Web.

5.3.1.1 - A arte na *Web*

O conceito de arte é divergente, subjetivo e varia de acordo com a cultura a ser analisada, com o período histórico ou até mesmo com um trabalho em questão. A arte na *Web* pode ser vista como a arte da leveza obtida do resultado de técnicas de uso de ferramentas gráficas e de design.

No início de todo o processo artístico ou científico existe desconhecimento e curiosidade bastante para se realizar trabalhos inovadores e viáveis, que ajudem o usuário a entender além da escrita pura e técnica.

A falta de conhecimento de educação artística e educação sensorial-visual faz com que haja um certo preconceito para com os trabalhos artísticos, na *Web*, de um modo geral. Essa falta de conhecimento é que faz com que algumas pessoas acreditem em mitos, na força do

conhecimento emocional, em metáforas, analogias, imagens que expressam situações surrealistas, narrativas e exemplos ilustrativos que não expressam uma forma de pensamento ordenado.

Reconhece-se que simplesmente um artista não é a melhor opção para se trabalhar com o desenvolvimento de *Web sites* (ou publicitário, jornalista, engenheiro, cientista etc, pois *design* para a *Web* é para *Web designers*), mas os trabalhos artísticos podem contribuir para uma estética melhor na fase do *design* de interfaces. O trabalho de um *Web designer* pode complementar o trabalho de um artista e vice-versa, assim como, um *Web designer* pode ter conhecimentos e habilidades artísticas (ou de outras profissões) para desenvolver melhores trabalhos.

O comportamento do artista é igual ao do Engenheiro e do *Web designer* quando ambos esperam pelos olhares de um público; por elogios; reconhecimentos etc. (Domingues, 2003).

Se um *Web site* for desenvolvido com base na engenharia, em técnicas de redação, de desenvolvimento e tratamento de imagens e também de conhecimentos artísticos, mais chances tem de aproveitar melhor o tempo e o lugar devido a cada etapa de um projeto.

A arte pela arte é difícil de ser empregada em obras de arquitetura, odontologia, engenharia, inclusive na engenharia *Web*. Porém a arte desenvolvida a partir de uma educação artística e de razões objetivas, podem representar a qualidade estética que diferencia um trabalho de outro; que faz com que um profissional seja mais requisitado e mais bem pago que outros; que um *Web site* seja mais visitado e bem sucedido que outros, mesmo em relação a outros que têm conteúdos semelhantes.

Muitas vezes o que ordena a arte na *Web* são as estratégias mercadológicas que buscam meios de agradar os usuários através de frases e ilustrações que despertam satisfações e curiosidades. O conhecimento de perfis de usuário e uma visão de possibilidades de agradá-los já não é mais papel da engenharia. A engenharia já cumpriu o papel no desenvolvimento e do bom funcionamento do *Web site*. Na fase de *design* das interfaces são conhecimentos complementares, de outras ciências, de outras técnicas e também de conhecimentos artísticos que representam a qualidade visual. A arte não atrapalha a engenharia, a engenharia não atrapalha a arte, desde que se dê a cada uma seu tempo e seu espaço buscando elevar a qualidade estética e funcional dos *Web sites* ao invés de torná-los "sem graça", simples e despersonalizados.

Sem conhecer um pouco de arte é difícil saber as impressões que as artes causam! E, isso dificulta algumas escolhas como, por exemplo, de uma imagem entre muitas semelhantes; a escolha de várias imagens para compor uma sobreposição; a escolha de várias cores para fazer composições e contrastes. A arte na *Web* proporciona percepções privilegiadas de entendimento intuitivo de um trabalho, tanto para o profissional que cria, quanto para os usuários que as percebem, que prestam a atenção a ela.

A criatividade pressupõe uma pessoa inventiva que produz e dá existência a "produtos" que não existiam anteriormente, ainda que neste produto haja uma contribuição parcial. A repressão acontece quando não são oferecidas condições criativas, e além disso, enfatiza-se o "não assumir riscos" e ficar no terreno seguro da repetição, da igualdade e do "já conhecido".

A arte é também usada na ficção, mas não quer dizer que a arte não represente a realidade de um fato. Através da imagem pode-se mostrar instâncias da realidade. Isso é feito, muitas vezes, com a ajuda de fotografias que representam o registro visual de um fato. No entanto, nem sempre é possível fotografar tudo o que ocorre, na realidade, na ficção, nos fenômenos naturais, restando a opção de registrá-los através de desenhos, sejam estes feitos em papel ou no computador (antigamente eram utilizados recursos como pedra, madeira, pergaminhos etc).

Mesmo as fotografias precisam muitas vezes, serem retocadas, compostas com outras, agrupadas em seqüência para simular animações etc. Para fazer estes registros, são necessários conhecimentos de educação artística, de formas, de curvas, de perspectivas, de inclinações, de sombras etc. Quanto mais habilidade tiver um profissional em registrar fatos reais ou "modelos mentais" criados para designar a ilustração de um fato, de uma sucessão de fatos, de um comportamento natural, mais chances existem de se passar informações que a escrita, por si só, não teriam a mesma precisão (Pauli, 1997).

Isso mostra que existe espaço para a arte na *Web*. Uma arte que é representada por uma caracterização de textos e imagens, *links* e formulários (sons, animações) com um *design* atraente, expressivo e com características de *design* planejado ao invés de *design* feito ao acaso. Existe espaço para a arte ficcionista utilizada em *Web sites* cujo motivo seja a ficção e a arte ordenada e precisa em *Web sites* que onde o objetivo é dispor ao usuário informações baseadas em acontecimentos cotidianos.

Os "*Web writters*" (bem como escritores de outros meios de comunicação) nem sempre são capazes de dizer "bem" o que pretendem dizer, assim como de ler, desenhar e escrever. Por outro lado, os usuários nem sempre são capazes de entender "bem" o conteúdo disponível nas interfaces (bem como em outros meios de comunicação).

Os símbolos gráficos representativos tornam-se espontaneamente acessíveis em "dizer mais". A comunicação se dá através sistemas de símbolos (escrita e imagem), como as formas de artes gráficas e múltiplas linguagens formais e coloquiais. A experiência virtual como um todo, não se traduz em uma forma exclusiva de símbolos, seja unicamente a escrita ou a imagem, ainda que descritos com base nas mais modernas técnicas discursivas e ilustrativas.

A educação para a arte na *Web* pode ser proposta pela atenção e pela convivência com outros *Web sites*, por leituras diversas sobre ética, estética e arte e pela técnica de uso das ferramentas gráficas e de *design*. Quanto mais ampla for essa convivência, mais chances haverá de se criar estilos atraentes e inovadores e que não comprometem o objetivo principal da *Web* que é o bom funcionamento navegacional e o rápido carregamento.

Em síntese, pretendeu-se expor o quanto os significados da arte na *Web* são tão básicos, que raramente há consciência deles, ou são tão estranhos quando seu estudo é ignorado, que a maioria dos escritores sobre *Web sites* e Engenharia *Web* não reconhecem que há espaço nem onde há espaço para a arte na *Web*; que a arte não atrapalha a engenharia, mas auxilia se usada no tempo e espaço que lhe é apropriado.

5.3.1.2 - A Estética na *Web*

A consciência estética (do mesmo modo que a consciência ética) é de ordem estritamente social e dificilmente existem se não estiverem em causa pelo menos duas pessoas (Pauli, 1997). Filósofos como e Alexander Gottlieb Baumgarten e Ludwig Wittgenstein é que abordaram a proximidade da ética e da estética, muitas vezes considerando-as como sendo um único assunto quando a ética e a estética representam a investigação geral sobre o que é bom (Moore, 1998).

A estética é um ramo da filosofia que aborda a essência e a percepção do belo e estuda racionalmente o belo e o sentimento que o belo suscita nos seres humanos. De acordo com a filosofia a estética é um conjunto de características formais que a arte assume em

determinado período e que pode, também ser chamado de estilo (Gallo, 1997), (Moore, 1998).

O termo Estética foi introduzido em 1753, pelo filósofo alemão Alexander Gottlieb Baumgarten, embora as primeiras teorias de certo alcance sejam as de Platão e de Aristóteles. Ambos falaram da arte como imitação da realidade e consideraram a estética inseparável da ética (Pauli, 1997).

A estética na *Web* pode ser entendida através de um usuário, que por meio de uma curiosidade ou de uma forma de acolhimento percebe possibilidades de significado de arte e "armazena" os significados contidos na tela. O resultado dessas percepções depende dos conhecimentos e do senso crítico de cada usuário: aos que não têm conhecimento de arte e de ferramentas de desenvolvimento a interface oferece a experiência de contemplar a estética; a sensação de estar diante do belo. Aos que são também criadores a experiência estética pode gerar os mais diversos tipos de indagação como, por exemplo, que ferramentas que foram utilizadas; que cores; que linguagem; qual a formação de quem criou; quanto tempo foi despendido para a realização daquele trabalho etc.

A diferença entre uma interface pura e simplesmente técnica e uma feita com arte, dedicação de mais tempo e mais zelo é a impressão que uma ou a outra podem causar. Um estímulo visual pode estimular a imaginação de forma a induzir ou sugerir alguma sensação aos outros sentidos (olfato, paladar, tato, audição), bem como despertar o interesse por determinados assuntos.

As câmeras fotográficas (principalmente as digitais) têm se encarregado de fazer grande parte dos trabalhos ilustrativos, mas a partir das imagens extraídas de câmeras há muito trabalho que pode ser refeito, retocado, recriado para proporcionar um diferencial estético. Os estilos de "realidade virtual", por exemplo, feitos a partir de fotografias estão entre os trabalhos mais valorizados na *Web* (indústrias de cosméticos, vestimentas, bijuterias, calçados e multimídias em geral).

Um dos princípios de usabilidade que fere princípios estéticos é a questão da "simplicidade", utilizada de forma extremamente enfática pelo escritor Jakob Nielsen, (Nielsen, 2000). Nielsen afirma que para ser bom tem que ser simples! Uma maneira de entender que isso nem sempre é verdade quanto se trata de interfaces de *Web sites* é quando um *Web designer* oferece a seus clientes um trabalho simples, qual é a chance de vender seu trabalho? Quando

se quer convencer um usuário a visitar um domínio não é falando que neste domínio tem interfaces com uma estética simples que se convence o usuário a fazer a visita.

Interfaces simples podem ser desenvolvidas por pessoas inexperientes que aprenderam a usar um editor qualquer de HTML (crianças, por exemplo, desenvolvem interfaces simples para fazer seus *blogs*). No entanto para fazer trabalhos com requinte, funcionais e com características de trabalhos provenientes de profissionais é necessário pessoas experientes, que conheçam os mais diversos aspectos tecnológicos e artísticos que envolvem os sistemas *Web*.

Ser simples, na *Web*, é diferente de ser bom. O que se pode buscar em interfaces requintadas e com aspectos profissionais é que além de requintada, funcional e profissional, ainda passe uma impressão de simplicidade, (quando essa noção de simplicidade signifique de entendimento simples, intuitivo, de entendimento popular, do vulgo, do povo).

O que se propõe para a questão estética, na fase de *design* das interfaces, é uma busca de que estéticas feitas com requinte, com zelo e com qualidade profissional para que sejam entendidas pelos mais diversos perfis de usuários. A partir daí pode dizer que é de uso simples, comum, mas não que a estética da interface seja simples ou comum.

A estética traz consigo uma espécie de marca técnica e artística de quem a produz. A *home page*, por exemplo, pode ser comparada à vitrine de uma loja onde o usuário (cliente) pode se interessar por algum *link* (produto) e acessá-lo (entrar na loja). Quais as probabilidades de uma loja vender seus produtos a partir de uma vitrine simples? São praticamente as mesmas de um *Web designer* vender trabalhos *Web* simples (vende, sem dúvida, desde que não haja ninguém da concorrência oferecendo trabalhos melhores).

A questão da estética das interfaces levanta questionamentos de como, por exemplo, em um cenário de pesquisas interdisciplinar como o que envolve o papel dos projetistas e artistas na pesquisa com interfaces humano-computador, onde a contribuição conjunta traria resultados melhores. Segundo, Domingues, a viabilidade desse tipo de colaboração é que os artistas tratam os problemas de uma forma bem diferente que os cientistas/engenheiros os tratam. A visão artística pode ampliar o processo de pesquisa definindo novos tipos de questões a serem pesquisadas; oferecer interpretações não-ortodoxas dos resultados; representar perspectivas potenciais do usuário e; colaborar de maneira geral com experiências artísticas-culturais para dar sustentação ao futuro de bases tecnológicas (Domingues, 2003).

5.3.1.3 - A Ética na *Web*

A ética é a investigação geral sobre o que é bom (Moore, 1998). É uma busca constante de fazer melhor aquilo que se faz. Filósofos como Ludwig Wittgenstein e Alexander Gottlieb Baumgarten (Pauli, 1997) mostravam estudos sobre a ética e a estética como sendo assuntos difíceis de serem separados considerando a ética como uma forma de consciência que busca fazer o melhor e a estética como uma concretização de um bem que foi feito (Moore, 1998), (Baldwin, 2003).

A ética das interfaces Web pode ser vista tanto em detalhes funcionais como na busca de um *design* mais atraente, dando ao *design* um tempo e um espaço para que se concretize sem comprometer a funcionalidade e a compreensão do conteúdo como um todo.

As redes globais de computadores dissolveram os limites geográficos. Uma sociedade interconectada eletronicamente e sem fronteiras é uma experiência relativamente nova que desafia as formas legais de diferentes países e de comunidades locais. Os problemas éticos e jurídicos são muito grandes e permanecem ainda amplamente insolúveis, pois necessitam de uma nova cultura, novas práticas e novas leis. E mesmo quando novas leis surgirem, servirão somente para julgar fatos onde tenha existido um agressor e uma vítima.

A falta de ética pode ser praticada não deliberadamente, mas devido a limitações tecnológicas ou inexperiência de profissionais que passam a exercer atividades de desenvolvimento de *Web sites* pelo simples fato de terem aprendido a usar uma ferramenta de desenvolvimento e não porque tenham adquirido qualificação profissional.

O que precisa ser considerado pelos profissionais da *Web* é que nem sempre alguns usuários fazem uso de sistemas de forma deliberada como, por exemplo, sistemas bancários, eleitorais, sistemas de compras, cadastros e inscrições *on-line* diversas.

Assim, a ética é abordada como uma possibilidade de ultrapassar as fronteiras dessa sociedade "sem fronteiras" que é a Internet e a *Web*, fazendo-se obediência ao que não é obrigatório: buscando as maneiras mais coerentes possíveis (dentro dos limites de cada profissional e dos recursos de cada projeto) para agradar e facilitar os procedimentos de quem pertence a essa sociedade. As interfaces *Web* são desenvolvidas para os usuários e usuários vêem e entendem o *design*, a arte. Conseqüentemente, é nas interfaces onde se encontra a necessidade de se ter mais cuidado com a forma e com a estética dos conteúdos disponíveis

aos usuários. Para o usuário não importa se a qualidade é proveniente da arte ou da engenharia; se é uma questão de ética, de estética, ou de usabilidade ou se todas estas questões é que representam um bom *design*.

5.3.1.4 - Considerações Técnicas, Artísticas, Estéticas e Éticas em Interfaces Web

A arte pressupõe uma técnica e a arte eletrônica pressupõe um uso sistemático de técnicas em ferramentas gráficas e de *design*. Enquanto um *Web designer* faz uso de técnicas e as aplica em ferramentas, pode-se dizer que se está diante um trabalho técnico. No entanto, o resultado do uso de técnicas, aplicado em ferramentas, pode ser visto como um trabalho artístico e criativo uma vez que com o uso das mesmas técnicas podem-se obter soluções diferentes, características da maturidade de cada profissional.

A seguir são mostrados alguns aspectos para os quais observa-se uma necessidade de atenção e de cuidados para que as interfaces sejam disponibilizadas ao usuário com estética e funcionalidade atraentes.

5.3.1.4.1 - Estrutura de uma Interface

A estrutura deve ser definida no inicio do projeto, pois é a partir desta que se desenvolve o sistema navegacional. No entanto é na hora da elaboração do *design* que os problemas de algumas estruturas mais aparecem. A estrutura de uma interface pode ser comparada à estrutura ou construção de uma casa. Primeiro levanta-se uma estrutura e depois se colocam os móveis e decorações dentro. Na *Web* desenvolve-se uma estrutura para colocar os atributos dentro. Uma estrutura pode ajudar ou dificultar a forma de navegação e a compreensão de uma interface. Alguns problemas mais freqüentes, provenientes do tipo de estrutura são mostrados a seguir.

- **Estrutura de frame**: exibindo a própria barra de navegação com uma página de terceiros na janela principal; dando a falsa impressão de que há uma página única na tela, mas na verdade, a tela fica dividida fazendo com que uma ação do usuário não seja respondida em todos os pontos da tela quebrando um princípio da *Web* que diz que "para cada janela do navegador, uma única página e um único endereço (URL)" (Baker, 2001); quando há acoplamento de janelas, levando à redução do tamanho do espaço disponível para visualização do conteúdo; quando a estrutura camufla o endereço de uma página no *location* do navegador, não permitindo que o usuário

mantenha o endereço em seus "Favoritos" ou anote para voltar mais tarde (Baker, 2001); quando gera barras de rolagem em pontos estratégicos da página. Estas barras geradas levam o usuário ao uso excessivo do mouse e comprometem a estética do *design* (Baker, 2001); quando fragmenta uma página, fazendo com que ela apareça sem o arquivo complementar que normalmente é a barra de navegação.

- **Estrutura de tabelas**: onde há excesso de repetição de código prolongando o tempo de carregamento de cada página e dificultando o trabalho do administrador; quando os arquivos estão em muitos níveis de subdiretórios fazendo com que os URLs fiquem muito extensos, dificultando o registro escrito dos mesmos.

5.3.1.4.2 - Coerência e Legibilidade dos Textos

Para que um texto seja coerente, ele precisa ser disposto com uma certa lógica ou uma seqüência planejada de acordo com cada conteúdo. Para ser legível tem que obedecer a algumas regras que facilitam a leitura, como tamanho da fonte, contraste da cor da fonte com a cor do fundo da tela, tipo ou *design* da fonte.

Problemas provenientes de textos podem ser vistos quando as cores das fontes têm um tom muito próximo da cor do fundo da tela; quando o tamanho da fonte é muito pequeno; quando o tipo da fonte é muito artístico ou serrilhado, fazendo com que uma letra se funda com a outra, causando dificuldade de leitura; quando existem ícones ou letrinhas seguindo o cursor do mouse, anulando o objetivo do cursor que é de dar parâmetro ao leitor do ponto onde ele está, na tela, e atrapalhando a visão do conteúdo (Baker, 2001); quando as formas de alinhamento não oferecem um parâmetro vertical como o conseguido com os textos justificados ou alinhados à esquerda.

É comum em *Web sites* a exibição de várias frases, que não ocupam a linha toda, centralizadas ou blocos de textos centralizados ou sem justificar. Isso faz com que não haja um ponto inicial de leitura, o que causa uma dificuldade maior para o usuário. Problemas de contraste aparecem principalmente quando são usadas cores que não fazem parte da tabela *Web safe* (216 cores). As cores que estão fora da tabela sofrem variações maiores, tanto quando usadas para cor de fundo como quando usadas para cor de fonte. O contraste visualizado em um tipo de navegador pode diferenciar muito do contraste visualizado em

outro. Isso faz com que reduza a precisão no contraste necessário entre cor de fundo e fonte, para que se tenha uma boa legibilidade (Baker, 2001).

5.3.1.4.3 - Uso de Metáfora e Termos Inadequados para a *Web*

Um outro problema muito comum pode ser visto nas abreviações, em misturas desnecessárias de idiomas e em usos inadequados de metáforas, títulos e termos representativos.

- **Abreviações**: as abreviações são necessárias algumas vezes. No entanto só devem ser usadas quando não há possibilidades de escrever a palavra toda, devido à falta de espaço. Se o espaço não é suficiente ao invés de utilizar várias palavras abreviadas, uma saída é tentar utilizar siglas (desde que em um espaço próximo e visível seja mencionado o seu significado); suprimir as palavras que não sejam de extrema importância (que não tirem o significado da frase) e substituir a frase por uma menor que tenha o mesmo sentido. Ao decidir que uma palavra será abreviada, então se abrevia a menor parte possível, utilizando a regra da direita para a esquerda (mantendo o início da palavra). No caso de frase, as abreviações são da direita para a esquerda (na palavra), começando as abreviações pelas primeiras palavras, mantendo as últimas (da direita) intactas. Para o usuário saber que é uma abreviação utiliza-se um ponto final após as letras da abreviação (como se faz para um texto impresso). O problema das abreviações e que quem faz a leitura tende a completar a palavra abreviada de acordo com uma frase ou palavra que vê com mais freqüência.
- **Misturas de idiomas**: há termos que não fazem parte de um idioma, mas que não têm um substituto preciso para que possa ser substituído. Essa é uma situação em que a mistura de idiomas é aceita. Um exemplo disso é uma barra de navegação ter uma opção chamada *links*; para uma sala de bate-papo um *link* com a palavra *chat* etc. Termos que sejam representativos de situações da *Web*, quando substituídos podem representar problemas ao invés de soluções. Mas, isso só vale para termos em que a palavra proveniente de outro idioma seja mais usada e defina melhor uma situação. Existe ainda a questão das frases incompletas como, por exemplo, o uso da palavra *home* que significa casa, para expressar *home page* que é a metáfora usada para expressar página inicial. O mesmo é observado no uso do *about* sem o complemento da frase, que pode ser "sobre a empresa, sobre o instituto, sobre mim, sobre um produto". Enfim, as frases incompletas podem ser intuitivas para usuários

experientes e um enigma para usuários em geral. Quando as frases (palavras) são duvidosas é necessário que o usuário acesse aquele *link* e se, tiver um título na página, poderá saber o que seria o significado. As frases incompletas também aparecem e representam um problema quando as palavras são do próprio idioma, no entanto são mais bem entendidas com palavras de idiomas diferentes.

- **Uso inadequado de metáforas, títulos e termos representativos**: as metáforas podem e devem ser usadas sempre que representarem uma informação de forma mais evidente que sua expressão natural. Uma metáfora pode ser representada pela seguinte equação: A = B. Quando A, que representa a situação real tiver menos eficácia de comunicação que B, então usa-se B, que é representação de A, dita com outras palavras como, por exemplo, *links* ao invés de "pontos (imagens ou textos) clicáveis" etc.

  Este exemplo mostra que praticamente não há problemas quanto ao uso de metáfora, pois estas representam uma situação real. Alguns problemas podem ser vistos quando as palavras ou frases não representam uma situação real, não são metáforas, mas representam aquilo que um profissional gostaria de dizer e não encontra palavras adequadas. Alguns exemplos são vistos em expressões como novidades, últimos lançamentos, o mais recente, o melhor, o mais etc. quando não se trata de um produto interno, que se aparecer uma outra novidade (ou um produto/serviço melhor) é somente proveniente do segmento ora representado pelo domínio. Se a suposta novidade for de um outro segmento onde novidades possam ser lançadas a cada dia, termos como estes ficam sem sentido. Uma opção é usar um termo representativo como, por exemplo, Pesquisas sobre o assunto x, *links* complementares, lançamentos da linha x etc.

  Quando em uma barra de navegação encontra-se o termo "*links*", o usuário normalmente acessa esperando encontrar uma interface de abertura que indica o endereço de outros domínios. O problema é quando, para representar esta situação são usadas frases como: hora do lanche, brindes etc. Termos como estes não representam metáforas e geram frustrações ao usuário que acessa esperando encontrar o que o *link* sugere.

5.3.1.4.4 - Peso do Montante de cada Interface

As imagens que fazem parte do *design* de um *Web site* são carregadas pelo navegador no momento em que um usuário as solicita. Muitas vezes uma imagem fala mais que muitas linhas de texto, portanto, quando bem relacionadas a um texto, elas não são dispensáveis ou inúteis, ao contrário, são um apoio, uma afirmação. Porém, se forem pesadas demais, ao invés de ajudar, elas poderão ser um obstáculo devido à demora do tempo de carregamento. Isso não quer dizer que os usuários tenham que ser privados das imagens de alta resolução, apenas que elas não precisam ser parte integrante do conteúdo, podendo estar dispostas para *download* em um ou mais tamanhos. Assim como as imagens, o excesso de texto em uma página também retarda o tempo de exibição de um conteúdo, levando o usuário ao excesso do uso da barra de rolagem.

5.3.1.4.5 - Harmonia na Distribuição dos Atributos que Compõem uma Interface

Uma interface normalmente é composta de textos, imagens e *links* (e formulários quando são interfaces dinâmicas). As imagens têm o objetivo de ilustrar o que foi dito em um texto e podem ser estáticas ou animadas. As imagens animadas servem para chamar a atenção do usuário para um assunto que mereça destaque naquela página. Os problemas aparecem quando existe um excesso de imagens em uma interface causando poluição visual, elevando o tempo para exibição e dificultando a mobilidade do usuário de um ponto para outro, devido ao peso da interface; quando se exibem mais de uma animação por página, deixando o usuário confuso sem saber a que dar mais atenção; quando as imagens são usadas sem necessidade, como por exemplo, onde um assunto já fala por si só. O uso indiscriminado de ícones em *links* é um dos maiores exemplos de uso desnecessário de imagens. Se há uma frase que explica o que é o conteúdo correspondente aquele *link*, não há a necessidade de colocar um ícone para ilustrar. Se um *link* leva o usuário a *home page*, não precisa de uma casinha para fazer uma ilustração e muito menos um envelope ou caixinha de correio para indicar um endereço de correio eletrônico (a maioria dos ícones representam signos do mundo real, se o usuário for leigo uma casinha representará sugestões peculiares, conforme a experiência de cada um). A falta de harmonia também é vista quando não há padronização da estrutura das interfaces, do uso de cores, fontes e do *design* em geral; quando há grandes diferenças do modelo de uma interface para outra, dentro do mesmo *site*, o usuário poderá ficar em dúvida se já saiu ou ainda está no mesmo *site* (Baker, 2001); quando não há

disponível uma barra de navegação (menu) para que o usuário possa saber onde está e para onde poderá ir (Baker, 2001) e quando não um caminho de volta quando o sistema de navegação vai além do segundo nível (Baker, 2001).

5.3.1.4.6 - Adoção de um Tipo de Linguagem e/ou Tecnologia

Estudar uma linguagem e fazer uso da mesma, é um procedimento básico de todo *Web designer* ou programador. Porém, quando se desenvolve uma aplicação para a *Web*, precisam ser levados em consideração alguns cuidados, principalmente quando se usam linguagens do lado do cliente, como JavaScript, CSS, XML etc. As linguagens do lado cliente dependem dos recursos da máquina do usuário para que o conteúdo de uma interface seja exibido. Por mais que se façam testes antes de publicar um *Web site*, não há um meio que saber que recursos existem na máquina de cada usuário. Isso só permite que o desenvolvedor tenha probabilidades de *design* e não certezas de que o conteúdo será visto na íntegra. Alguns problemas decorrentes da linguagem/tecnologia podem ser vistos no uso de:

- ***FullScreen***: o *FullScreen* (tela cheia) utiliza toda a tela do computador fazendo com que o usuário perca a barra de ferramentas do navegador (Baker, 2001).
- **Excesso de *plug-ins***: com tantos recursos para desenvolvimento de páginas *Web*, não há necessidade de utilizar tecnologias que façam com que o usuário tenha que baixar *plug-ins* para visualizar o conteúdo de uma interface, ainda mais se esse *plug-in* tem poucas perspectivas de ser utilizado para visualização de outros *Web sites* (algumas exceções são os *plug-ins* de Java e Flash, que são bastante utilizados na *Web*).
- **Substituição dos atributos originais**: a caracterização dos atributos originais das interfaces representa um *design* personalizado, mas a substituição destes atributos pode gerar algumas duvidas ao invés de facilidades. Alguns exemplos podem ser vistos quando são usados botões de comandos para a navegação ao invés de serem usados para fazer a ativação de páginas dinâmicas (Leite, 2002); quando o sublinhado dos *links* é retirado etc.

5.3.1.4.7 - Objetividade

Dificilmente um usuário entra em um *Web site* com o objetivo de navegar, somente pelo motivo de ficar clicando em *links*. Um usuário clica em um *link* para encontrar o que a frase exposta no *link* sugere. Alguns problemas de objetividade podem ser vistos em situações como:

- **Insuficiência de dados**: nas interfaces onde existem alguns dados disponíveis, mas o usuário não tem informações sobre o que eles são, como podem ser usados e porque estão lá. Essas páginas dão a impressão de que aquele fragmento de conteúdo foi "jogado" e esquecido lá (o *link* ocupa um lugar nobre na barra de navegação, mas ninguém sabe ao certo qual é o seu objetivo).
- **Nas páginas *merchandising***: onde um *link* sugere um conteúdo informativo, mas na verdade leva o usuário a propagandas disfarçadas no meio de textos com conteúdo diferente do sugerido no *link*.
- **Nas páginas de abertura sem conteúdo**: que fazem com que o usuário acesse uma página de abertura e aí simplesmente tenha que clicar em um *link* para entrar definitivamente na página. Há situações mais extremas, onde o usuário tem que escolher uma versão, de linguagem ou tecnologia, para entrar na página (Baker, 2001).
- **Nas páginas em construção**: onde só há uma mensagem de que algo está "Em Construção", mas não fala o que está sendo construído e nem a data que estará pronto para o usuário voltar à página se tiver interesse pelo assunto.
- **Nas informações inúteis**: alguns exemplos de informações inúteis, que não proporcionam nada ao usuário e ainda ocupam um lugar de destaque em uma página, representam falta de maturidade por parte do desenvolvedor ou administrador. Essas informações aparecem na forma de contadores de acesso visíveis, algo que só interessa ao administrador do *Web site* e não ao internauta; na forma de ordens sem objetivo, como por exemplo, "Clique Aqui" ao invés de um *link* com uma frase referente ao que existirá se o usuário "Clicar" no *link*.
- **Na falta de democracia**: a falta de democracia é vista em informações de exclusividade ou de restrição tecnológica, como por exemplo, resolução

recomendada, navegador recomendado, excesso de *plug-ins* e até recursos de *hardware* necessários (Baker, 2001).

- **Nos conteúdos desatualizados**: que podem induzir o usuário a erros, em uma tomada de decisão; na falta de informações necessárias como a última data de atualização de cada página (Baker, 2001);
- **Na falta de critérios de disponibilidade dos assuntos**: como, por exemplo, por ordem de importância e quando há vários assuntos com a mesma importância, uma seqüência por ordem alfabética ou de acordo com uma hierarquia referente ao assunto. Isso vale tanto para *links*, principalmente em menus, como para segmentações de textos e páginas (Baker, 2001).

5.3.1.4.8 - Canais de Comunicação

Os canais de comunicação como *e-mail*, formulários de cadastro, *chats*, listas de discussão, *instant message*, são recursos que fazem com que se o usuário quiser mais informações do que as disponíveis no *site*, saberá a quem contatar e de que forma. Esses canais devem estar visíveis em todas as páginas como uma forma de afirmar que alguém é responsável pelas informações disponíveis e poderá oferecer informações adicionais caso o internauta as solicite (Baker, 2001), (IEEE, 2001).

No entanto há situações em que os canais de comunicação são usados de forma excessiva e arbitrária como, por exemplo, os formulários quando aparecem no carregamento de uma interface em forma de caixinhas de alertas que não permitem que o usuário dê seqüência na navegação enquanto não preenche os campos de dados e clica em um botão de envio; quando aparecem no carregamento de uma interface em janelas *pop-up*; quando ficam em forma de *fly* sobrepondo o conteúdo da interface.

Muitos tópicos ainda poderiam ser abordados, sem que esgotassem os espaços onde a arte, a ética e estética (ou a ausência destas) podem ser observadas como benefícios para o usuário. Não como uma lei, como mandamentos ou padrões de *design* a serem seguidos, mas como observações de um conjunto de características que podem ser usadas em um determinado tempo e espaço, revelando, assim, o estilo do *Web site*. Este estilo é que representa um diferencial na qualidade do *design*, que é o que efetivamente é visto pelo usuário.

Ao concluir a fase de design das interfaces os arquivos precisam ser enviados novamente ao servidor e isso requer novas avaliações e novos testes, ainda que já tenham sido feitos ao concluir a fase de implementação.

5.3.2 - Avaliações e Testes

Após a conclusão da fase de *design* das interfaces, aparecem tipos de testes que não foram realizados ao final da fase de Implementação. Embora os testes navegacionais já tenham sido feitos a partir da máquina cliente e da máquina servidora, após terem sido feitas mudanças, na fase de *Design* da Interfaces, é necessário que o sistema de navegação seja testado novamente, considerando que o sistema agora já está disponível *on-line*. Neste estágio a ajuda de usuários é de grande valia, pois muitas vezes a equipe de desenvolvimento já está tão familiarizada com o *Web site* que erros básicos (digitação, descrição de imagens, frases de *links*, endereços de *links* etc.) podem passar desapercebidos. Alguns tipos básicos de testes que precisam ser feitos nesta etapa tanto pela equipe de desenvolvimento e administração e se possível por usuários diversos são avaliações visuais, testes de navegação e testes de tempo de carregamento de cada interface.

- **Avaliações visuais**: no final do desenvolvimento de um projeto, as interfaces normalmente já são intuitivas para a equipe de desenvolvimento. Porém é necessário considerar que o *Web site* foi desenvolvido pela equipe, mas não para a equipe e sim para o usuário. Então, um forma de avaliação é solicitar a usuários com conhecimentos diversos que façam um *tour* pelo *site* e forneçam um *feedback* com considerações como: as interfaces são intuitivas? Encontrou com facilidade o que o *site* sugere dispor? Encontrou questões antiéticas? O *design* está atraente, original e de acordo com os motivos do domínio? Onde e como o *Web site* poderia ser melhorado?

- **Testes de navegação**: nesta fase, os testes de navegação precisam ser feitos a partir da máquina servidora por diversos navegadores; diversos sistemas operacionais; diversos sistemas de banda, seja larga ou estreita. As diferenças de *design* entre navegadores, sistemas operacionais e sistemas de banda são comuns. Quando se usa uma linguagem que utiliza um servidor de páginas dinâmicas como PHP, por exemplo (usada neste processo), essas diferenças são bastante reduzidas ou quase eliminadas, pois é o servidor que se encarrega de gerar o conteúdo e não os recursos

do cliente. Como há também linguagens que dependem dos recursos da máquina cliente como, CSS e JavaScript etc., os testes de navegação são indispensáveis para que haja uma certificação de que as diferenças não comprometem o entendimento do *design* e do conteúdo do *Web site*.

- **Peso e tempo de carregamento**: testes feitos a partir de diferentes tipos de banda representam as respostas finais referentes a manter ou fazer modificações de *design*. O que pode reduzir o tempo de carregamento de uma interface é a quantidade de texto e de imagens; a peso das imagens e o tipo de editor utilizado (formatação automática ou codificação manual). Ainda que as avaliações de *design* estejam de acordo com o objetivo do domínio, os testes com o tempo de carregamento (*download*) podem revelar necessidades de redução do espaço ocupado por uma imagem para que possa haver uma maior compactação; redução do número de imagens por interface; segmentação do conteúdo gerando novas interfaces; escolha de um outro editor (quando do uso de formação automática).

  Uma outra questão quanto ao tempo de carregamento de uma interface deve ser considerada em relação ao tipo de usuário predominante ou da segmentação mercadológica. Se o objetivo do *Web site* são os acessos de um perfil de usuários onde a maioria dispõe de recursos atualizados de *software* e *hardware* e conexão por banda larga os resultados considerados razoáveis são diferentes dos resultados para um perfil mais diversificado ou predominante de recursos mais modestos.

CAPÍTULO 6

RESULTADOS

O processo proposto já foi aplicado em *Web sites* de alguns domínios e outros que foram desenvolvidos somente para a realização de testes.

As aplicações foram feitas com o uso de ASP e PHP e somente alguns testes para estrutura dinâmica foram feitos com JSP. O processo proposto foi implementado PHP, conforme mostrado na fase 2 e pode ser feito, seguindo os mesmos procedimentos, também em ASP.

Alguns resultados obtidos através da utilização do PDW-UML podem ser vistos em alguns exemplos, como os relacionados a seguir.

- **Implementação de um novo tipo de estrutura feito com arquivos-matriz**: a estrutura formada pelos arquivos-matriz propõe e mostra melhorias tanto para a equipe, ao reduzir o trabalho de implementação como para o usuário que tem estruturas padronizadas, seqüências exatas em menus etc. O uso de arquivos-matriz representa uma contribuição para a engenharia ao representar uma forma de implementação formalizada e testada. As estruturas propostas até então representam recursos de uso do HTML (*frames* e tabelas) que podem ser usados da maneira que o desenvolvedor considerar melhor. A estrutura de *frames* reduz o trabalho para o desenvolvedor, mas representa muitos problemas para o usuário. A estrutura de tabelas oferece as mais diversas possibilidades de uso, sendo inclusive um recurso utilizado tanto em estruturas com *frames* como pela estrutura de arquivos-matriz, no entanto, a estrutura em si não propõe um uso formalizado que represente benefícios à equipe e aos usuários. Se um desenvolvedor optar por usar estruturas de arquivos-matriz, o fará a partir do conhecimento de como desenvolver um *Web site* com este tipo de estrutura e isso já representa obter o benefício das melhorias pertinentes à estrutura.
- **Redução dos níveis de URLs**: o modelo de Estrutura Dinâmica (ED) proporciona ao desenvolvedor e administrador a organização da distribuição dos arquivos e diretórios, conforme seja necessário para facilitar o entendimento do sistema, independente de quantos níveis sejam necessários. Para ter esta mesma organização nos demais modelos (Seção 4.1), o usuário precisa acompanhar toda a estrutura

lógica existente na máquina, o que resulta em URLs muito longos, difíceis de serem memorizados e trabalhosos para serem anotados. O modelo ED consegue solucionar o problema dos URLs longos fazendo com que os usuários acessem os arquivos somente de dois níveis que é o nível *root* e um abaixo do *root* onde ficam os arquivos-matriz. Todos os demais arquivos constantes em outros diretórios e subdiretórios têm papel de "servidores de conteúdo" para os arquivos-matriz não representando níveis de URLs para os usuários.

- **Abordagem de conceitos**: o PDW-UML aborda alguns conceitos e mostra onde são utilizados em *Web sites* mostrando alguns exemplos do que é uma estrutura de dados e uma estrutura de interface, o que é um *design* e o que é um atributo que compõem o *design* de uma interface. Aborda ainda o papel da engenharia e o papel filosófico (arte, ética e estética) dentro de um *Web site*, mostrando que ambos têm seu tempo e seu espaço sem que um anule ou atrapalhe a função desempenhada pelo outro.
- **Uso da UML e de novos estereótipos**: o uso de estereótipos propostos para representar as interfaces auxilia o desenvolvimento de novos projetos sem que haja a necessidade de fazer descrições dos tipos de interfaces, pois o estereótipo já faz a representação. O uso dos diagramas da UML mostrando atributos das interfaces e as necessidades do projeto também colabora para proporcionar uma linguagem comum aos *Web designers*. Como os estereótipos são parte integrante do PDW-UML, o desenvolvedor que conhecer o PDW-UML conhecerá também os estereótipos e poderá utilizá-los em seus projetos, desenvolvendo-os em qualquer editor de imagens, pois os mesmos não apresentam recursos que sejam de um editor específico.
- **Valorização do trabalho artístico e criativo**: a proposta de uma fase para o *design* dentro de um processo procura valorizar o trabalho artístico e criativo mostrando que sem a engenharia e sem um projeto formalizado a funcionalidade pode ser comprometida. Por outro lado, a partir de um trabalho feito dentro de padrões da engenharia ainda há viabilidades para um tratamento no *design* onde a arte e a criatividade podem proporcionar uma qualidade estética não prevista pela engenharia.

- **Redução do tempo de *download*:** o sistema de reutilização de arquivos, conforme feito com as diretivas de *include* de SSI, , representam uma redução de tempo de *download* devido à estrutura formada pelos arquivos-matriz. Quanto mais arquivos estiverem incluídos em outros arquivos onde há reincidência de chamadas, menos códigos o sistema terá que interpretar e menos tempo levará para concluir o *download* de uma interface.

- **Reutilização de conteúdo**: existem várias maneiras de fazer uso de SSI, de CSS, tabelas, *frames* etc. No entanto, a maneira como SSI e demais recursos são utilizados é que se conseguem os resultados mostrados como melhorias de usabilidade para o usuário e reusabilidade para a equipe. Conforme a implementação do PDW-UML também se separam algumas categorias de arquivos, facilitando o desenvolvimento e a reutilização. Dessa forma é possível reaproveitar os arquivos para desenvolver novos *Web sites* e para reaproveitamento no próprio *Web sites* em caso de mudança de tecnologia.

  - Arquivos de interface: os arquivos de interface trazem instruções de características de *design* (são os arquivos com instruções de CSS, com design de textos, tabelas etc.) podendo ser aproveitados em outros *Web sites* e com outras tecnologias, bastando fazer pequenas modificações como, por exemplo nos códigos das cores. Esses arquivos podem ser utilizados em interfaces que não usem estrutura dinâmica, fazendo inclusão com diretivas de *include* de CSS.

  - Arquivos-matriz: como os arquivos-matriz fazem somente a chamada de outros arquivos, podem ser modificados para uso em uma tecnologia diferente como, por exemplo, de PHP para ASP e *vice-versa*, bastando fazer modificações das diretivas de *include* conforme a linguagem.

  - Arquivos de conteúdo: os arquivos de conteúdo têm a mínima utilização de instruções de código, (os códigos estão nos arquivos de interface) por isso podem ser reutilizados com outras tecnologias sem a necessidade de grandes modificações.

CAPÍTULO 7

CONCLUSÃO

O que se pretendeu com o presente trabalho foi mostrar um processo que envolva o levantamento de requisitos, a implementação e o *design* procurando colocar em prática os objetivos da engenharia ao buscar usabilidade (facilidade de uso) para o usuário e reusabilidade (facilidade de usar novamente) para a equipe.

A facilidade de uso pode ser vista na estrutura padronizada das interfaces. Independente de como seja o *design* final, as interfaces têm a mesma estrutura e a mesma forma de organização. Isso reduz as possibilidades de que o usuário perca a ordem da navegação ou tenha dúvidas se já esteve naquela interface ou se ainda está no mesmo domínio o que normalmente ocorre quando não se mantém um padrão de estrutura, seja estrutura de interface ou estrutura de dados. A facilidade de reutilizar pode ser vista na reutilização dos arquivos e por não ter a necessidade de repetições de código ao fazer inclusões de arquivos de forma ordenada para obter o máximo de reaproveitamento e o mínimo de repetições.

Este trabalho não representa o desenvolvimento de uma nova tecnologia, mas apresenta o uso conjunto de algumas linguagens e tecnologias existentes mostrado em um processo de desenvolvimento. Este processo propõe estereótipos específicos para interfaces de *Web sites*; apresenta abordagens diversas que podem facilitar o desenvolvimento de novos projetos como, por exemplo, o levantamento e a análise dos requisitos para *Web site* e propõe uma forma de documentação que pode servir de *help* ou guia para novos profissionais e para os que já estejam envolvidos em um projeto.

Algumas diferenças entre a engenharia de *software* e a engenharia *Web* podem ser vistas na fase de implementação (Seção 5.2) onde as formalizações utilizadas representam soluções para a *Web*, mas teriam poucas possibilidades de aproveitamento ou uso para *softwares* convencionais. Para o desenvolvimento de *softwares* convencionais não há a necessidade de um planejamento da distribuição e nomenclatura de arquivos e diretórios com foi proposto para a *Web*. A documentação proposta para *Web site* também é diferente da documentação de *softwares*. Enquanto a documentação de *software* está mais voltada ao uso do *software*, a documentação do *Web site* está mais voltada para registros de informações, convenções e apenas uma pequena parte voltada à orientação de como trabalhar com o *Web site*. Na

primeira fase as diferenças podem ser vistas na forma de levantamento e seleção dos requisitos. As semelhanças são mostradas principalmente na modelagem e nas possibilidades de uso da UML para ambos os casos.

Ambas têm pontos em comum, mas finalidades diferentes, fazem uso de recursos diferentes e têm usuários diferentes. Ainda que as fases e 1 e 2 sejam representadas por soluções provenientes da engenharia, a fase 3 representa um tempo e um espaço para o trabalho artístico e criativo. Assim podem ser vistas algumas diferenças entre a arte e a engenharia e que ambas são necessárias ao desenvolvimento de *Web sites*: a engenharia faz uso de formalizações e soluções já testadas e a arte mostra que é possível uma solução diferente de acordo com os objetivos de cada domínio.

O processo apresentado neste trabalho mostrou a implementação feita em PHP e faz referências também a ASP e JSP. Implementações em ASP já foram feitas e apresentam praticamente os mesmos resultados obtidos com o uso de PHP quanto ao projeto, forma de implementação e tempo de *download*.

O que ainda não foi implementado são aplicações em JSP. Normalmente quando há aplicações desenvolvidas em Java, estas representam trabalhos mais complexos onde há um grande volume de dados e conseqüentemente, bancos de dados e geração de interfaces dinâmicas.

Como perspectiva de um novo trabalho estuda-se a possibilidade de desenvolvimento de uma ferramenta que tenha um editor de códigos, que entenda a dinâmica do sistema, a hierarquia formada pelas diretivas de *include* e um sistema de desenvolvimento que crie os diagramas para a modelagem. Uma ferramenta com estas (e outras) características poderia ser uma forma de auxílio para os programadores gerenciar o projeto como um todo.

Se os programadores, através do uso de uma ferramenta que entenda a dinâmica do sistema, conseguissem entregar um *Web site* já implementado aos *Web designers*, para que façam o tratamento do *design* como proposto pela fase 3 do PDW-UML, poderia ser um bom meio de integração entre profissionais. A integração através do uso de uma ferramenta poderia representar o uso de um propósito comum que é garantir a qualidade e o funcionamento do sistema nas duas primeiras fases e finalizar o trabalho com o tratamento do *design*.

REFERÊNCIAS BIBLIOGRÁFICAS

Aranha, M. L. **Filosofando**: introdução à filosofia. São Paulo: Moderna, 2003. 440p.

Baker, A. **Theory**. 2001. Disponível em: <http://www.merges.net/theory/>. Acesso em: 20 nov. 2004.

Baldwin, T. **A hundred years of principia ethica**. 2003. Disponível em: <http://www.cfh.ufsc.br/ethic@/ETH@21~1.PRN.pdf>. Acesso em: 20 nov. 2004.

Bell, Ian. **HTML, DHTML & Web design**. São Paulo: Market Books, 2000. 617p.

Booch, G. et al. **UML**, guia do usuário. Rio de Janeiro: Campus, 2000. 472p.

Bos, B. **Cascading style sheets**. 2003. Disponível em: <http://www.w3.org/Style/CSS/>. Acesso em: 20 nov. 2004.

Br.php.net. **Manual do PHP**. 2003. Disponível em: <http://br.php.net/tut.php>. Acesso em: 20 nov. 2004.

Bruce R. S. The Interspace: concept navigation across distibuted communities. **Computer-IEEE**, v. 35, n. 1, Jan. 2002, p. 54-62.

Ceri, S. Fraternali, P. Bongio, A. Modeling data entry and operations in WebML. In: International Workshop on the Web and Databases, 3., WebDB 2000, Dallas, USA, 2000. **Proceedings...** Dallas: ACM PODS/SIGM, p. 201-214, 2000. Disponível em: <http://www.research.att.com/conf/webdb2000/PAPERS/6b.pdf>. Acesso em: 20 nov. 2004.

Ceri, S. Fraternali, P. Bongio, A. Web modeling language (WebML): a Modeling Language for Designing Web Sites. **Computer Networks-The International Journal Of Computer And Telecommunications Networking** , v.33, n.1-6, p. 137-157, Jun. 2000. Disponível em: <http://www.webml.org/webml/page1.do>. Acesso em: 20 nov. 2004.

Chase, N. **Aprendendo active server pages 3.0**. São Paulo: Makron Books, 2000. 406p.

Chauí, M. **Convite à filosofia**. São Paulo: Ática, 2000. 440p.

Conallen, J. Modeling Web applications architectures with UML. **Communications of ACM**, Oct. v. 42, n. 10, p. 63-70, 1999.

Domingues, D. **Arte e vida no século XXI**: tecnologia, arte e criatividade. São Paulo: Editora UNESP, 2003. 380p.

Drăgănescu, M. **Broadband Internet and the knowlwdge society**. 2003. Disponível em: <http://www.racai.ro/~dragam/BROADBAND-INTERNET-AND-TKNOWLEGDE.pdf>. Acesso em: 20 nov. 2004.

Franklint, K. **ASP**, Técnicas e Estratégias. São Paulo: Érica, 2001. 346p.

Gallo, S. **Ética e cidadania**: caminhos da filosofia. Campinas: Papirus, 1997. 79p.

Gardner, H. **Inteligências múltiplas**, a teoria na prática. Porto Alegre: Artmed Bookman, 1997.

Garzotto, F.; Schwabe, D.; Paolini P. HDM - A model based approach to hypermedia application design, **ACM transaction on information systems**, v. 11, n.1, p.1-26, Jan.1993.

Graef, G.; Gaedke, G. Development and Evolution of Web-Applications using the Web-Composition Process Model. In: International WorkShop on Web Engineering at International Conference, 9. , (www9), 2000, Amsterdam. **Proceedings**… Amsterdam: TecO, 2000. Disponível em: <http://www.teco.edu/~gaedke/paper/2000-www9-webe.pdf>. Acesso em: 20 nov. 2004.

Gudwin, R. **Linguagens de programação**. Campinas: DCA/FEEC/UNICAMP, 1997. Disponível em: <http://www.eng.uerj.br/~araujo/disciplinas/Caract/ling_prog.pdf>. Acesso em: 20 nov. 2004.

Haddleton, F.; Pfaltz, J. L. Client/Server Architecture in the ADAMS Parallel Object-Oriented Database System, In: International, Conference on Scientific Computing in Object-Oriented Environments, ISCOPE '97, 1., 1997, Marina del Rey, CA, **Proceedings**… Marina del Rey: Disponível em: <http://www.cs.virginia.edu/~adams/iscope.pdf>. Acesso em: 20 nov. 2004.

Hurlburt A. **Layout: o design da página impressa**. São Paulo: Nobel, 2002. 157p.

Isakowitz, T. Stohr, A. Balasubramanian, P. RMM: a methodology for structured hypermedia design. **Communications ACM**, v. 38, n. 8, p. 34-44, Aug.1995.

Isaak, J. **Web site engineering best practice standards** (IEEE 2001), 2001. Disponível em: <http://standards.ieee.org/reading/ieee/std/internet/2001-2002.pdf>. Acesso em: 20 nov. 2004.

Instituto Brasileiro de Opinião Pública e Estatística (IBOPE). **Parceria entre IBOPE e ACNielsen eRatings.com**. 2004. Disponível em: <http://www.ibope.com.br/>. Acesso em: 20 nov. 2004.

Kirda, E. Jazayeri, M. Kerer, C. Experiences in engineering flexible Web services. **IEEE Multimedia**, v.8, n. 1, p. 58-65, Jan-Mar 2001.

Leite, J. **Design e usabilidade de sistemas Web**. 2002. Disponível em: <http://www.dimap.ufrn.br/~jair/diuweb/>. Acesso em: 20 nov. 2004.

Lima, m. **Considerações em torno do conceito de estereótipo**: uma dupla abordagem. 1986. Disponível em: <http://sweet.ua.pt/~mbaptista/consideracoes%20em%20torno%20do%20conceito%20de%20esterotipo.pdf>. Acesso em: 20 nov. 2004.

Ministério da Ciência e Tecnologia (MCT). **Internet e telecomunicações**. Brasília, MCT, 2004. Disponível em: <http://www.mct.gov.br/temas/info/imprensa/internet.htm>. Acesso em: 20 nov. 2004.

Medeiros, M. **Conhecendo ASP, PHP e JSP**. 2002. Disponível em: <http://nuclinfo.famato.org.br/down/poster_AI_02.pdf>. Acesso em: 20 nov. 2004.

Mendes, M. A. **Comparação e análise de requisições dinâmicas em servidores Web**. 1999. Disponível em: <http://www.dcc.ufmg.br/pos/html/spg98/anais/corelio/corelio.html>. Acesso em: 20 nov. 2004.

Mesquita, R. C. **Analise e projetos orientados a objetos**. 2003. Disponível em: <http://www.ead.eee.ufmg.br/~renato/appoo/>. Acesso em: 20 nov. 2004.

Microsoft Corporation. **Internet information services**. 2003. Disponível em: <http://www.microsoft.com/WindowsServer2003/iis/default.mspx>. Acesso em: 20 nov. 2004.

Microsoft Corporation. **Using VBScript and jscript on a web page**. 1998. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnvid/html/msdn_vbnjscrpt.asp>. Acesso em: 20 nov. 2004.

Moore, G. **Principia ethica**. 1998. Disponível em: <http://www.rbjones.com/rbjpub/philos/bibliog/moore03.htm>. Acesso em: 20 nov. 2004.

Netscape Communications Corporation (NCC). **Cliente-side JavaScript guide**. v. 1.3, 1999. Disponível em: <http://plbpc001.ouhk.edu.hk/%7Emt834/course-units/unit-8/materials/mt834_unit8/ClientGuideJS13.pdf >. Acesso em: 20 nov. 2004.

Nielsen, J. **Projetando websites**. Rio de Janeiro: Campus, 2000. 362p.

Novaes, A. **Ética**. São Paulo: 9. ed. Companhia das letras. 2003. 395p.

Pauli, E. (ed.), **Enciclopédia Simpózio**: história da filosofia moderna, cap. 15. 1997. Disponível em: <http://www.cfh.ufsc.br/~simpozio/novo/2216y605.htm> Acesso em: 20 nov. 2004.

Pereira, A. Schwabe, D. **Especificação declarativa de aplicações Web em OOHDM**. 2001. Disponível em: <http://www-di.inf.puc-rio.br/schwabe/papers/Adriana%20SBMidia%202001. pdf>. Acesso em: 20 nov. 2004.

Prestwood, M. **Static versus dynamic content.** 2003. Disponível em: <http://prestwood.com/internet/design/default.asp>. Acesso em: 20 nov. 2004.

Ribeiro, R. J. **Códigos de ética**. 2004. Disponível em: <http://noticias.aol.com.br/colunistas/renato_janine/2004/0030.adp>. Acesso em: 20 nov. 2004.

Roldão, D. **Estudo sobre inteligência artificial**. 2000. Disponível em: <http://www.citi.pt/educacao_final/trab_final_inteligencia_artificial/>. Acesso em: 20 nov.

Roselli, A. **Usable web**: frames. 1999. Disponível em: <http://usableweb.org>. Acesso em: 20 nov. 2004.

Rossi, G.; Schwabe, D.; Barbosa, S. **OOHDM**: Object-Oriented Hypermedia Design Method. Tese de doutorado, PUC-Rio, 1996. Disponível em: <http://www-lifia.info.unlp.edu.ar/~fer/oohdm/allindex.htm>. Acesso em: 20 nov. 2004.

Secretaria de Política de Informática (SEPIN). **Internet comercial**: evolução da Internet no Brasil e no mundo, conceitos, estatísticas e aspectos legais. Brasília: MCT, abr, 2000.

Shimbun. K. Nihon. **Nihon Keizai Shimbun Incorporation**. Disponível em: <http://www.nikkei.co.jp>. Acesso em: 20 nov. 2004.

Sun.com. **Sun one Active Server Pages**. 2003. Disponível em: <http://wwws.sun.com/software/chilisoft/>. Acesso em: 20 nov. 2004.

The Apache Software Foundation. **Apache HTTP server project**. 2002.Disponível em <http://httpd.apache.org>. Acesso em: 20 nov. 2004.

Yager, Tom. **ChiliSoft gives a ASP sizzle to Unix**. v.21, n.17, Apr. 1999. Disponível em: <http://www.infoworld.com/cgi-bin/displayArchive.pl?/99/17/i02-17.47.htm>. Acesso em: 20 nov. 2004.

Zafiris, A. P. A Practitioner's Approach to Evolving and Remodeling Large-Scale WWW sites. In: Hawai International Conference on System Sciences, 34., Jan. 3-6, 2001, Maui, Hawai. **Proceedings**… Maui, Hawai: IEEE, 2001. Disponível em: <http://csdl.computer.org/comp/proceedings/hicss/2001/0981/07/09817078.pdf>. Acesso em: 20 nov. 2004.

Zelenovskiz, R. & Mendonça, A. **Funcionamento de modems**. Jun. 2001. Disponível em: <http://www.gabrieltorres.com.br/modems.html>. Acesso em: 20 nov. 2004.

W3C, traduções. **W3C em sete pontos**. 2003. Disponível em: <2002. http://www.amtechs.com/w3c/w3c7points.html>. Acesso em: 20 nov. 2004.